National Library of Canada | Bibliothèque nationale du Canada

CANADIAN THESES ON MICROFICHE | THÈSES CANADIENNES SUR MICROFICHE

NAME OF AUTHOR/*NOM DE L'AUTEUR* MARGARET MARIE SHARON

TITLE OF THESIS/*TITRE DE LA THÈSE* A SIMULATION STUDY OF ~~FF~~ SCHEDULING STRATEGIES FOR LOAD SHARING COMPUTER NETWORKS

UNIVERSITY/*UNIVERSITÉ* U. OF ALBERTA

DEGREE FOR WHICH THESIS WAS PRESENTED/*GRADE POUR LEQUEL CETTE THÈSE FUT PRÉSENTÉE* M.Sc

YEAR THIS DEGREE CONFERRED/*ANNÉE D'OBTENTION DE CE GRADE* 1975

NAME OF SUPERVISOR/*NOM DU DIRECTEUR DE THÈSE* DR. J. TARTAR

DATED/*DATÉ* August 15/75 SIGNED/*SIGNÉ* Margaret M. Sharon

PERMANENT ADDRESS/*RÉSIDENCE FIXÉ* c/o Mrs. G D Sharon
3956 Yew Street
Vancouver BC V6L 3B6

THE UNIVERSITY OF ALBERTA

A SIMULATION STUDY OF SCHEDULING STRATEGIES FOR LOAD SHARING COMPUTER NETWORKS

by

MARGARET M. SHARON

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTING SCIENCE

EDMONTON, ALBERTA

FALL, 1975

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled "A Simulation Study of Scheduling Strategies for Load Sharing Computer Networks", submitted by Margaret Marie Sharon in partial fulfilment of the requirements for the degree of Master of Science.

.....
Supervisor

.....

.....

.....

Dated ...August.11..... 1975.

Abstract

The increasing development and use of resource sharing computer networks motivates investigation into methods for scheduling jobs and resources in a network, as opposed to single processor scheduling. A number of scheduling strategies based on various job and device characteristics are proposed. These strategies are analysed and compared with the aid of a load sharing network simulation.

Acknowledgments

I am grateful to Dr. John Tartar for his guidance and constant support as my thesis supervisor.

To the Department of Computing Science and the National Research Council I extend my appreciation for financial assistance provided.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (cont'd)

CHAPTER I

INTRODUCTION

The topic of this study is an investigation into strategies for the allocation of resources and jobs in a computer network, specifically for the purpose of sharing the total computing load among the network members.

Extensive research and development is currently under way in the field of computer networks. A significant contributing factor has been the rapid growth of the related technology, especially data transmission facilities. Computer networks are now being used in many and varied applications in business and education.

The term "computer network" has been used to refer to various kinds of single-computer and multi-computer systems. For this study a computer network is defined as a group of self-contained computer installations interconnected by a communications network for the purpose of exchanging information or sharing resources. A communications network consists of communication lines and the hardware and software used to control data or message transmission. Remote job entry systems are not classified as computer networks as the interface processors are used only to transmit data to and from the remote devices. Similarly, multiprocessing systems differ from computer networks in that they share common memory and are not geographically distributed.

A primary objective of the linkage of computers into a network is the mutual sharing of the aggregated resources. The concept of resource sharing incorporates both the hardware resources of the network (CPU's, memories, peripherals, etc.) and the software resources (applications programs, data bases, etc.). The use of network facilities may reward the user with a wider range of services, increased computing power and decreased turnaround time. Improved throughput of jobs and optimal device utilization are among the goals of resource sharing from a management point of view.

The scheduling of a network's jobs and resources is accomplished in a manner analogous to that of an operating system in a single processor computing system. In the case of a network the multiplicity and accessibility of resources increases the complexity of the scheduling problem. Goals such as improved job throughput and optimal resource utilization will overlap, requiring scheduling decisions based on numerous different factors.

> "Computer network [scheduling] disciplines will also have to be dependent on transmission delays of service requests and jobs or parts of jobs from one computer to another as well as on the possible incompatibilities of various types between different computers. The synthesis and analysis of ... multiple processor network priority disciplines remains a fertile area of research..." [C3 p.20]

A brief survey of some aspects of current research in computer networks with an emphasis on resource sharing is

presented in Chapter II. Chapter III comprises an analysis of resource sharing strategies, extensions of them and the development of new strategies. A computer network simulation program was written to conduct experiments on the proposed strategies. A description of the simulation experiments and the results obtained are given in Chapter IV. Conclusions based on the results, with directions for further research, are discussed in Chapter V.

CHAPTER II

SURVEY OF RESEARCH IN COMPUTER NETWORKS

Four aspects of current research in computer networks are surveyed in this chapter. The issues of network structure, composition, communications and resource sharing are discussed. In lieu of a comprehensive enumeration of existing networks and their characteristics, examples for each of the four issues are taken from significant research or actual networks. Resource sharing in particular is emphasized. The ARPANET, Distributed Computing System (DCS) and two simulation studies are reviewed in the section on resource sharing.

2.1 Structure

A computer network may be represented by a graph in which the graph's nodes correspond to the computers (which are often referred to as hosts) and its edges to the communication facilities between them. The structure of a network is the arrangement of the nodes and their connecting links.

A network whose structure is centralized consists of a number of satellite computer systems, each of which is linked to a central system. (This configuration is also known as a star.) The major functions of network control are resident in the central system and it serves as the inter-

mediary for communication between the satellite systems.

The Octopus network in the Lawrence Berkeley Laboratory of the University of California [FR1,M2] was designed as an experimental centralized network. The central node was given the responsibility of managing a shared data base, concentrating and distributing incoming messages and controlling input and output resources. The reliability of the communication channels, the processor and memory at the central node proved to be crucial factors in the network's performance. When hardware and software modifications were being implemented, the interdependence of the components caused instability in the network.

As the centralized structure accounted for these difficulties, the Octopus network was redesigned with a distributed structure composed of two interconnected but independent sub-networks between which the network control is distributed. The two functions of shared data base management and remote terminal service are now controlled by separate processors. Alternate data paths and a dual processor at one node have been incorporated into the network for reliability.

A particular type of distributed structure called a ring is used in the Distributed Computing System at the University of California, Irvine [FR3]. Each host is connected into the network via a hardware device known as a ring interface (RI). Each RI is linked to two others,

forming a communications ring. A fixed number of message slots circulate in the communications ring from one RI to another.

The DCS network is a small (seven-node) experimental network developed as a tool for the study of distributed computing and distributed file systems. Its ring structure is not likely to be economical for large networks, due to its restricted number of message slots and few direct inter-host links.

2.2 Composition

Network composition is concerned with the processors at the hosts. A **homogeneous** network is one in which the processors at all hosts are of identical types and a **heterogeneous** network consists of hosts with different processor types. It is readily apparent that a heterogeneous network poses difficult problems in interprocessor communication.

The TSS network [W2,A1 p.458] consists of a number of IBM 360/67's, each using the TSS/360 operating system. Terminal users at the nodes may issue TSS or network commands to be respectively interpreted by their local host or relayed to the remote host for which they are intended. Consistency in communication between the TSS systems is achieved by the local hosts appearing as terminals to the remote hosts. The Computer Access Method acts as an interface with the communications hardware and establishes the

communications protocol, the prearranged sequence of events in the dialogue between the systems. The exchange of programs and large amounts of data is accomplished economically and modifications to the network are straightforward due to the uniformity of the nodes.

Incompatibilities between implementations of assemblers and compilers cause problems in homogeneous networks in which the component processors have different operating systems. Conversion of data structures presents another problem. However, these difficulties are minor when compared with those involved in heterogeneous networks, wherein lie fundamental differences in machine architecture. Complex hardware and software interfaces are needed to permit communication between the foreign machines.

The availability of specialized resources and access to desired data are two advantages of heterogeneous networks. An example in point is the ARPANET [FR1,R1] which consists of a multitude of processor types at over forty sites spanning the United States. A valuable member of the ARPANET is the ILLIAC IV with parallel processing and mass storage capacity of a trillion bits [A1 p.498]. The interconnection for data exchange between ARPA research centres was one of the original motivations in the development of the ARPANET.

Effective means are required to aid communication between heterogeneous hosts in networks. This problem is discussed further in the next section.

2.3 Communications and Protocols

Communication channels in networks are of two types: circuit-switched and message-switched. A circuit-switched network establishes a dedicated communication path between two communicating members and transmits all information via this connection. Message-switching networks route messages on the network's communication links using intermediate nodes between the source and destination [K1].

Packet switching, a method of segmenting long messages for transmission, has evolved with the development of the ARPANET [C2]. Messages are divided into fixed-length blocks which are each given the destination address and a sequence number before being routed onto any of the possible paths between the source and destination nodes. (There are at least two paths between any two nodes in the ARPANET.) The dismantling and reassembling of messages as well as the choice of paths is performed at the nodes by minicomputers known as Interface Message Processors (IMP's). The advantages of packet switching include minimal transmission delay of messages, increased accuracy, an even distribution of message traffic and increased utilization of communication lines.

Optimal routing strategies will attempt to avoid congestion and deadlock and optimize the flow of messages through the network. Current research using methods of

simulation and mathematical analysis to develop optimal routing strategies [K1] will facilitate the design of future networks.

Fredericksen [F1] has examined some solutions for communication problems caused by differences in the data descriptions of heterogeneous processors. Each applications program could be equipped with the necessary information for conversion of data from any form in which it may arrive to that which the program requires, but this suggestion is rejected as being too rigid. A standard universal data description language which would be comprehensible by all processors in the network is proposed. Each file of data to be transmitted would be preceded by a description of its structure, characteristics and contents, all coded in the data description language. A mechanism for translation into the recipient processor's language would be provided in the form of a utility or applications program at each node.

Each computer network appears to have its own protocols for communication. Two examples taken from a recent paper on this subject [A2] illustrate the diversity of methods.

Messages inserted in the communications ring of the DCS [A2,FR3] are addressed to processes or broadcast to classes of processes rather than to specific processors. Thus the sender need not know the location of the recipient, and processes may be resident at one or more of the nodes. Each RI checks each incoming message to determine whether it is

addressed to one of its active local processes. If so, it copies the message and sets a message-received bit in it before passing it on. If no node receives the message it is allowed to circulate in the ring a number of times prior to its removal by the sending node.

Walden [A2,W1] has proposed a protocol for a distributed network which uses a rendezvous table at each node, containing entries for unanswered SEND or RECEIVE messages received at the node. A SEND message is originated by a process which wishes to transmit information to another process and a RECEIVE is sent by a process which is prepared to receive information from another process. SEND and RECEIVE messages have five parameters: the SEND and RECEIVE ports (the data paths to and from the processes), the rendezvous site (the node to which the SEND or RECEIVE message is to be sent), a description of the buffer in which the message is to be sent or received, and the restart address of the process which initiated the SEND or RECEIVE message.

Upon receiving a SEND message a process replies with a corresponding RECEIVE message or vice versa. The matching SEND or RECEIVE message is deleted from the rendezvous table at the rendezvous site. SEND and RECEIVE messages, if unmatched for too long, are deleted from the rendezvous table and the originating process is notified.

Standardization of communications protocols may become

a major issue of future research. The interconnection of multiple computer networks will be one of the motivations for protocol standardization [M3].

2.4 Resource Sharing

Resources which are to be allocated in a resource sharing network include programs and data as well as the hardware devices. If jobs are transferred from one host to another it is for the purpose of obtaining specialized services or data. A job in a **load sharing** network is transferred to provide economical service by distributing the total workload among the network members. Load sharing with a view to equalizing the utilization of resources and evenly balancing the workload is known as **load leveling.** **Process scheduling** is the act of maintaining a queue of waiting processes in some order of priority and dispatching the most eligible process by allocating to it a processor and any other needed resources. The following sections focus on three approaches to resource sharing, as exemplified in descriptions of the Resource Sharing Executive of the ARPANET, the Request for Quotation scheduling method of the DCS, and two simulation studies of scheduling algorithms for resource allocation.

2.4.1 ARPANET

The Resource Sharing Executive System (RSEXEC) functions as a subsystem of the TENEX operating system on PDP-10 hosts of the ARPANET [T1]. Features of TENEX include a virtual processor and memory and an interactive command language. An ARPANET user at one of these hosts may avail himself of local and remote resources, including files at other TENEX hosts. The local RSEXEC uses an established protocol to interact via the independent communications system with service programs of the RSEXEC's at remote TENEX hosts. The service programs keep in contact with each other by exchanging status information periodically.

The use of remote resources is achieved by issuing commands which cause subsequent use of device names (e.g. LPT - line printer) to default to particular devices at specific hosts. A user profile, maintained by the RSEXEC at the local host, consists of a group of directories of files accessible to the user at each remote TENEX host.

The RSEXEC system provides for greater cooperation between members of the ARPANET by removing from the user the burden of direct interaction with remote hosts. It should be noted that all resource sharing is initiated explicitly by the user. Implicit forms of resource sharing such as load sharing are not included in the current RSEXEC. Further studies are to explore the extension of the RSEXEC to support of resource sharing among heterogeneous hosts [T1].

2.4.2 Distributed Computing System

A policy for resource allocation called the Request for Quotation (RFQ) method is employed by the DCS [FR2,FR4]. A process needing remote resources of a particular kind broadcasts a "request for quotation" by issuing a message in the communications ring to the appropriate class of processes, called resource allocators. A resource allocator at each node returns to the requestor a bid for the needed service, determined by some algorithm. After a fixed length of time the requesting process evaluates the submitted bids and sends an acknowledgment message to the resource allocator at the chosen node. To prevent resource allocators from bidding on services that they may not be able to supply and to prevent processes from gaining the use of too many resources, a central file is kept, containing limits to the amount of resources that resource allocators can supply and processes can receive. When a resource allocator receives an acknowledgment of its bid, both it and the requesting process inform a process known as the notary which ratifies the contract between them after checking the limits in the central file. The name of a process created thereafter by the resource allocator to represent the service given is transmitted to the requestor.

A Distributed File System has also been implemented in the DCS using the RFQ policy for creation of new files and the allocation of space for new versions of existing ones. A

requesting process locates a file by broadcasting the owner's name to all nodes, each of which contains a copy of the central component. The central component is a table of owner names and corresponding processes called catalogs, which access all files belonging to each owner. When the requesting process receives the catalog name from the first node to respond it sends a message to that catalog. The catalog returns the name of a volume process which handles access to the location at which the file resides. The volume process receives the next message from the requestor and, if protection requirements are met, causes a process to be created and its name (corresponding to the file name) to be passed to the requestor.

The Distributed File System and the RFQ method are distributed, reliable and flexible systems. Distributed control requires processes in need of resources to inform the resource allocators, as the latter would otherwise be forced to query all nodes in the network, seeking processes which need their services. As no central authority is involved, the failure of any component in either system does not seriously hinder the operation of the system. Of critical importance in the success of either system and requiring further examination, is the determination of efficient algorithms for computing the resource allocation bids.

2.4.3 Simulation Studies

Two simulation studies of resource sharing are reviewed in this subsection. The objective of the first research was to test algorithms for assigning job priorities and performing load leveling in a three node network. In the second study an excessive use of network resources by remote jobs was noted and strategies to control this occurrence were examined.

Bowdon [BW1,BW2] proposed a priority assignment scheme for jobs, in which the user specified a deadline for completion of his task. The dispatcher at each node attempted to process as many jobs in the node's process queue ahead of their deadlines as possible, thereby achieving the maximum reward for the use of the resources.

Waiting jobs were assigned priorities in each processing queue by a static or a dynamic evaluation method. The static assignment scheme computed a priority for each job based on a function of the user's estimates of CPU time, memory and the number of I/O requests needed to complete the job. The dynamic assignment scheme altered the priorities of jobs as a function of the job's age and the current system load and stability. A job's priority increased with the approach of its deadline or with its ability to restore balance if the CPU or memory resource was not being sufficiently utilized. Highest ratio of CPU time to number of I/O requests was the criterion for the job to be chosen

to improve CPU utilization. Memory utilization was increased by dispatching the job with the largest core request. Load leveling was performed periodically in the system by transferring a percentage of the jobs in the processing queue of the busiest node to the processing queue of the least busy node.

Results of experiments showed a decrease in job turnaround times at a node when either of the priority assignment policies was used. The dynamic allocation scheme produced an increase in turnaround time over that of the static scheme but resource utilization was improved. The inclusion of a load leveling algorithm resulted in decreased turnaround time at busier nodes, an increase in total throughput of jobs in the network and better resource utilization.

Helander [H1] observed, in a simulation of a network with no resource allocation control, the occurrence of a phenomenon called fragmentation. Fragmentation was defined as a network condition in which "the majority of resources in use belong to remote jobs" [H1 p.46]. This condition was deemed undesirable due to "increased communications costs... [and]... processing time to redistribute jobs" [H1 p.50]. Two control strategies were proposed: limited request and limited acceptance. The limited request policy allowed jobs to use remote resources only if the local node could not supply the needed resources. The limited acceptance policy

introduced at each node a threshold which constrained the amount of resources which could be allocated to remote jobs. Fragmentation was eliminated with the application of each strategy, and the limited acceptance strategy produced the least utilization of remote resources.

The foregoing resource sharing policies of DCS, Bowdon and Helander are to be further discussed in the next chapter. An extension of some of their characteristics and the development of other strategies will be presented.

CHAPTER III

STRATEGIES FOR RESOURCE AND LOAD SHARING

The previous chapter has surveyed the research in computer networks with particular attention to the resource sharing policies of the Distributed Computing System, Bowdon and Helander. These three studies will be analysed and compared in this chapter. An application of Helander's limited acceptance strategy will be introduced and two original strategies, based on transmission cost and memory requirement of jobs, will be developed.

3.1 Analysis of Scheduling Strategies

Each of the three studies employs a different method for determining the order in which jobs or resources should be evaluated for scheduling. The relative merits of each method deserve mention.

Jobs on the wait queue of Helander's simulation [H1 p.94] are examined as candidates for execution according to a polling system. In any sampling interval, jobs awaiting processing are appended to the wait queue as they arrive, and the selection of a job to be processed (termed the "feasible initiation event") is performed as follows:

(1) The first (i.e. oldest) job on the wait queue is examined. If enough resources are available at its local node it becomes the feasible initiation event.

(2) If not the network is polled in search of the first node with sufficient resources. If one is found the job becomes the feasible initiation event.

(3) If no node can process this job, the procedure in (1) and (2) is repeated with succeeding jobs on the wait queue until a feasible initiation event is found or the wait queue is exhausted.

This algorithm guarantees that the oldest job whose resource request can be immediately satisfied will be dispatched. However, if more than one remote node was capable of processing a job, there is no guarantee that the first node found could most efficiently process this particular job.

Depending on the measure or measures of network performance which are judged to be most important, this algorithm may or may not give optimal results. As jobs are selected in order of decreasing age, this algorithm is likely to result in low turnaround times. More efficient use of resources could have been obtained (though at the expense of greater turnaround time) had the nodes with available resources examined the eligible jobs. A given node would then have had the opportunity to select from the entire wait queue the job or jobs which would best utilize its resources.

The shortcoming of the polling method is that it bases its evaluation (whether of resources or jobs) on incomplete and biased information. With the bidding policy of DCS all available nodes of the network have an equal chance of being

assigned to a waiting job. A bidding system has greater potential than a polling system and can be implemented with almost no central control [FR2 p.543]. However, broadcasting requests and evaluating submitted bids would probably consume more time than would be taken to query the nodes sequentially.

Bowdon's priority assignment scheme [BW2] ordered jobs in the wait queues of each node independently. To accomplish load leveling a number of jobs were periodically transferred from the busiest node to the least busy node. With this exception, each node processed only its own jobs and did not communicate at all with other nodes or execute other node's jobs. This scheme is clearly less efficient than the previous two, due to the near absence of resource sharing.

A degree of centralization of control is used in each of the three systems. Bowdon's system assumes the existence of a central authority, to receive from each node the statistics of its utilization and to direct the transfer of jobs for load leveling. Jobs to be processed in Helander's simulation are contained in a single wait queue. DCS employs a central file of limits of the nodes' resources. For reliability reasons however, it is advantageous to distribute control of the scheduling process as much as possible.

Determination of the correctness of a simulation is a difficult task. With reference to this, a few remarks on the operation of the two simulation programs follow.

Bowdon's simulation program was designed to model a three node network and was patterned after an actual network. The results obtained by the simulation are restricted, due to the network size and small number of experiments conducted. However, the determination of values for parameters used in the priority assignment formula was aided by comparison of the performance of simulation runs with the performance of the real network under the same conditions. In addition, the validity of the simulation could be estimated by comparison of results obtained from the actual network and those obtained from the simulation.

In Helander's simulation priority for execution was given to local nodes but fragmentation occurred nonetheless. This phenomenon may be a consequence of the design of the simulation program. As explained at the beginning of this section the wait queue was examined, seeking the first job which may be processed by any node (if a job's local node could not process it). If each node with available resources had scanned the wait queue for local jobs that it could process, less processing of jobs by remote nodes would have occurred and significantly different results may have been produced.

Values of network throughput and turnaround times were not given, so no comparison could be made of these measurements under different strategies. Resource utilization was the most crucial factor but consideration of the effects of

the strategies on other performance criteria is necessary for completeness. Nonetheless, the limited acceptance strategy has considerable potential and an application of it will be developed in the next section.

3.2 Proposed Scheduling Strategies

Each of the three subsections which comprise this section is concerned with a scheduling strategy for load sharing. The first section deals with an application of Helander's limited acceptance strategy with some modifications, and two bases for bidding strategies are developed in the latter two subsections.

3.2.1 Fragmentation Control

Fragmentation is caused by insufficient constraint on the amount of a host's resources which may be allocated to remote jobs. Utilization of remote resources however, is resource sharing - a fundamental property of networks. It should be noted that it is excessive remote utilization, causing decreased throughput and increased turnaround, which is undesirable and which necessitates efficient scheduling strategies.

Constraint on remote usage of resources can be achieved by the limited acceptance method described in Chapter II. If all batch jobs in a load sharing network are required to use the output devices of the same host at which they were input, then use of these resources by remote jobs will not

occur. A limited acceptance constraint would therefore have to be placed either on the use of the CPU or on the storage media. Storage devices are more naturally partitioned among an arbitrary number of jobs than is CPU power. In particular, the memory at a node is easily segmented. For this reason the limited acceptance threshold will be considered to be a portion of the memory size at a node. This will restrict the amount of memory which can be allocated to remote jobs for execution or occupied by remote jobs which have executed and are awaiting transmission for output at their local node.

The use of resources by remote jobs may also be constrained by a strategy of giving priority to local jobs. Local jobs are awarded the first opportunity to use a CPU or any available memory they may need for retransmission from a remote node.

The use of this strategy in combination with the limited acceptance strategy might be more effective than the use of either strategy alone. These two strategies could be applied at each execute queue with a number of operating system scheduling algorithms: first-come-first-served or shortest job first [B1 pp.206-209], for example.

3.2.2 Least Transmission Cost

If transmission times of remote jobs constitute a significant fraction of their total turnaround time, then a

strategy based on least cost of transmission may be advantageous. Application of this strategy would result in increased utilization of the resources at those nodes associated with communication lines of the greatest capacity. These nodes could be the ones with the most powerful processors or most storage. This strategy is biased toward maximization of total network throughput and job turnaround times, but not necessarily toward optimal resource utilization at all nodes.

The least product of a job's size (in memory units) and the transmission rate between its local node and any remote node (in memory units per time unit) among all jobs, will determine the job to be selected. If the cost of transmission between a node and itself is considered to be zero, then this strategy will favour local jobs. As described above, it is possible to employ other criteria such as first-come-first-served for the selection of a local job from the local execute queue.

3.2.3 Least Memory Requirement

Memory at a node in a load sharing network will be occupied by remote jobs as well as local jobs. Limited acceptance restricts the amount of memory that remote jobs may use. Memory allocation will also be controlled by keeping the amount of memory in use evenly balanced at each node. Such a strategy would attempt to ensure that nodes with less storage would not become backlogged with local

jobs awaiting execution and/or remote jobs awaiting retransmission to their local node for output. An equal distribution of the use of resources is promoted by this algorithm. Turnaround times and throughput of jobs may be improved, but this would not be directly caused by the application of this strategy.

The job dispatched will be the one for which the ratio of its memory requirement to the memory available at a node is least. This strategy also favours local jobs, as they already occupy some of the memory at their local nodes when being considered for scheduling.

3.3 Summary

This chapter has focused on scheduling strategies implemented or simulated in three research studies. An analysis of each has shown some of the features and shortcomings of each one. From this analysis, one strategy has been modified and two new strategies have been proposed.

The strategies developed in the previous section will be tested using a simulation program, as will be described in Chapter IV. The design of the simulation will incorporate the bidding scheme for evaluation of jobs and resources. Three performance measurements will be included: network throughput, turnaround time and resource utilization. This will enable close examination of the properties and the potential of each strategy.

CHAPTER IV

SIMULATION EXPERIMENTS AND RESULTS

This chapter commences with a brief description of simulation as a model for the behaviour of a computer network. The need for simplifications and restrictions in the simulation of a real network are discussed, as are the various indicators of the network's performance. An approximate measure of the load of a simulated network is derived by means of queuing theory analysis. Two series of experiments were performed, each series comprised of two sets.

The first series is an investigation of fragmentation. The extent of remote utilization is compared with observed measurements of network performance, and experiments using methods of controlling fragmentation are compared. A smaller load factor is used in the second set of experiments of the first series, in order to check consistency with the results obtained in the first set.

The second series of experiments also consists of two sets, each having different load factors. This series compares three scheduling strategies: the least transmission cost (LTC) strategy, the least memory requirement (LMR) strategy, and the first-come-first-served (FCFS) strategy with fragmentation control which resulted from the first series. A summary of results obtained from both series of experiments concludes the chapter.

4.1 The Simulation Model

Simulation is a useful tool for modeling the behaviour of single-computer and multi-computer systems. Events occurring in the real-life system are traced in the simulation model, thereby revealing in detail the dynamic behaviour of the actual system. By gathering statistics of the operation of the system during the simulation, conclusions about performance can be drawn. Each simulation run may be considered to be an experiment by which results are obtained from observations and conclusions are made. Changing the values of parameters in the simulation for different experiments produces different sets of results for comparison with previous ones.

Some simplifications and constraints have been incorporated in the simulation, due to the vast number of parameters which characterise an actual computer network. Details of the design and operation of the simulation are to be found in the Appendix.

Observations of three parameters are displayed in graphs of the simulated network's performance using different scheduling strategies. The first performance measure is network throughput. The ratio of the number of jobs processed to the number of jobs generated during the simulation has been taken as a convenient measure of throughput. This is equivalent to the accustomed throughput

measure of the number of jobs processed per unit time, as the total simulated time was the same for all simulation experiments, and identical job streams were generated for all experiments having the same network size. Throughput is the primary measure of performance, as the main objective of a load-sharing network is to distribute the combined workload from all the hosts in such a manner that the network as a whole processes more jobs than the hosts working independently can process.

Average turnaround time of jobs, of vital importance to the user, is the second measure. The turnaround time of a job is defined as the time interval between the job's arrival and its completion of output. Average turnaround time differs from network throughput as the processing time of one job, which can be calculated from the network throughput, does not allow for the concurrent execution of jobs on some or all of the nodes.

Resource utilization is the third measure, defined as the portion of the total simulated time in which the resource was busy. In the case of memory utilization, resource utilization is measured as the average portion of occupied memory during the simulation. Average CPU utilization has been chosen as the measure of resource utilization, as input and output devices are used on a first-come-first-served basis by their local jobs only.

An important parameter in the experiments is the

measure of the load on a simulated network. The load factor is obtained by considering the simulation as a queuing theory model, as explained below.

A computer network as modelled by this simulation, may be described in queuing theory terminology as a system of queues in a three phase series. The first and third phase correspond respectively to the input and output devices and queues, and consist of N independent single servers, each having its own queue. The second phase, corresponding to the CPU devices and execute queues, is a multiple server with N channels, the queue of which is the composition of the execute queues at the N nodes of the network. It is to be noted that each CPU processes each job serially, as multi-programming is not a property of the simulation. The input, execute and output times of jobs in the simulation are equivalent to the service times of the devices. A Poisson process with parameter λ_1 describes the generation of jobs at each node (i.e. their arrival at the input devices). The service times of input, CPU, and output devices are exponentially distributed with means μ_1, μ_2 and μ_3 respectively. The choice of probability distributions to describe arrival and service times is discussed further in the Appendix.

Figure 1 shows the configuration of the queuing system. The diagram shows jobs from all CPU devices being appended to all output queues. In fact the output queue which a job

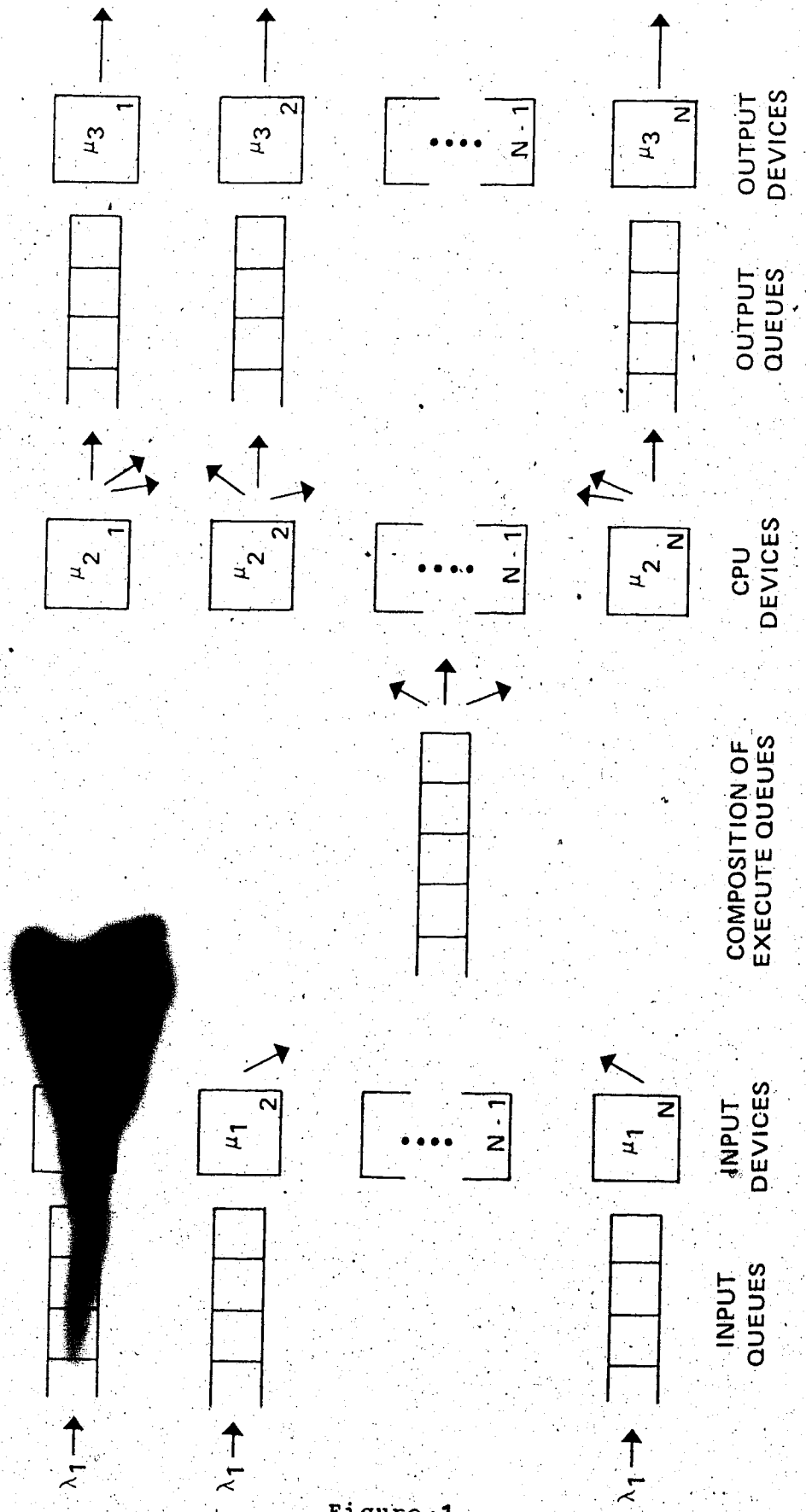

Figure 1
The Computer Network Simulation as a Queuing System

joins is predetermined, as jobs must output on the same node at which they were input. The arrival rate of jobs at the output phase is dependent on their arrival and servicing rates at the CPU phase, and this knowledge is used to derive the load factor.

The departure rate from a server at which jobs arrive as a Poisson process with parameter λ and are serviced according to an exponential distribution with parameter μ, is itself described by a Poisson distribution with the same parameter λ. This result holds only if a queue of unlimited length is allowed and the mean service rate is greater than the mean arrival rate [S1 pp.255-256]. The ratio of mean arrival rate to mean service rate (λ/μ) is called the utilization factor, and indicates the mean number of jobs which arrive during the mean interval in which a job is being serviced [B1 p.201].

The departure rate of jobs from the input devices is Poisson-distributed with parameter λ_1, as the input processing phase has Poisson arrival rate of jobs, an exponential service rate (the input rate of jobs) greater than the arrival rate, and a potentially infinite queue. The distribution of the departure of jobs from the input devices becomes the distribution of arrivals at the CPU devices. As there are N independent input devices the arrival of jobs at the CPU phase is a Poisson process with parameter $N\lambda_1$. The CPU phase consists of N independent devices, each having an

exponentially distributed service time (the execution time of jobs) with parameter μ_2. The overall service rate of this phase is exponentially distributed with parameter $N\mu_2$.

The load factor (ρ) for the simulation is defined to be the ratio of the mean arrival rate of jobs at the CPU phase and the mean service rate at this phase.

$$\rho = \frac{\text{mean arrival rate}}{\text{mean service rate}} = \frac{N\lambda_1}{N\mu_2} = \frac{\lambda_1}{\mu_2}$$

The load factor has no dimensions and is independent of N, the network size.

The mean execution rate of jobs in the simulation was chosen to be 0.0167. (Information on the choice of input parameters is to be found in the Appendix.) Two values for the mean arrival rate of jobs at input devices were used in separate sets of experiments in the fragmentation control and strategy comparison series. A value of 0.0182 for the mean arrival rate produces $\rho_A = 1.09$, and a value of 0.0167 gives $\rho_B = 1.0$. In both cases, the mean arrival rate of jobs is greater than or equal to the mean service rate, which indicates that the network load will increase as time advances in the simulation.

Preliminary experiments were performed to find suitable values for the mean arrival rate. These experiments showed that the two values given above for the mean arrival rate of jobs produced the most acceptable performance statistics,

and did not cause the network to become overloaded during the fixed period of time simulated.

4.2 Results of Simulation Experiments

Two series of simulation experiments were conducted. The purpose of the first was to investigate the phenomenon of fragmentation, its effects and methods for its control, and of the second to compare the two strategies based on transmission cost and memory requirement (which were described in Chapter III) and the best fragmentation control strategy obtained in the first series. As mentioned in the previous section, each series of experiments was composed of two sets, the second set having a different value for the network load factor, to confirm whether or not the results were consistent with those of the first set. The load factor in each first set of experiments was 1.09 and in the second set 1.0.

4.2.1 Fragmentation Control

The limited acceptance threshold and local priority algorithms were employed as scheduling strategies with the first-come-first-served algorithm in experiments of networks with sizes ranging from three to nine nodes. The first-come-first-served scheduling algorithm [B1 pp.206-207] gives priority to the job with the earliest arrival time, without regard to its other characteristics. In the first set of experiments limited acceptance thresholds of 5, 10, 18, 25,

50 and 100 percent of the memory size at each node were used as examples, testing the entire range to discover the thresholds most likely to give optimal results. Each of these thresholds was tested with and without the local priority strategy. It is to be noted that a limited acceptance threshold of 100 percent of the memory size is equivalent to no threshold, enabling examination of the local priority algorithm (and its absence) without any limited acceptance threshold.

Figure 2 shows the correspondence between remote utilization and throughput for the experiments in this set. Remote utilization is determined as the ratio of the number of remote jobs processed to the total number of jobs processed during the simulation. The graph indicates the increase in throughput as remote utilization increases from 0 to approximately 20 - 40 percent. As remote utilization increases beyond this level throughput decreases and fragmentation occurs. In all cases the experiments in the latter range lacked the local priority (LP) strategy, as shown by the symbol which denotes them. Optimal throughput corresponds to approximately 10 percent remote utilization and these experiments used the LP strategy.

The percentage of throughput achieved with the limited acceptance (LA) thresholds for different network sizes is shown in Figures 3a and 3b. Figure 3a shows the results for LA thresholds of 5, 10 and 18 percent and Figure 3b shows

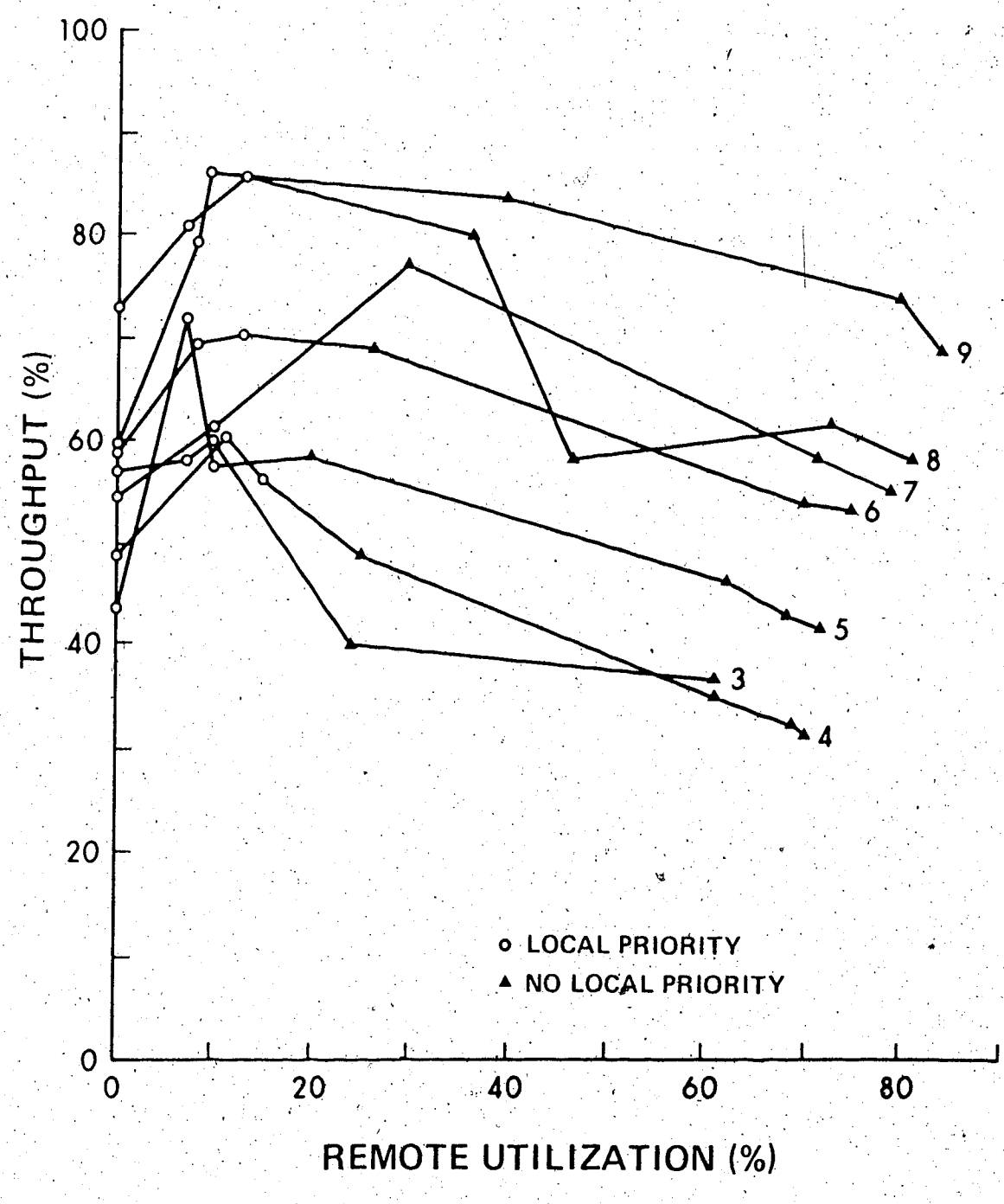

Figure 2
Throughput by Remote Utilization
(Fragmentation Control Series, First Set)

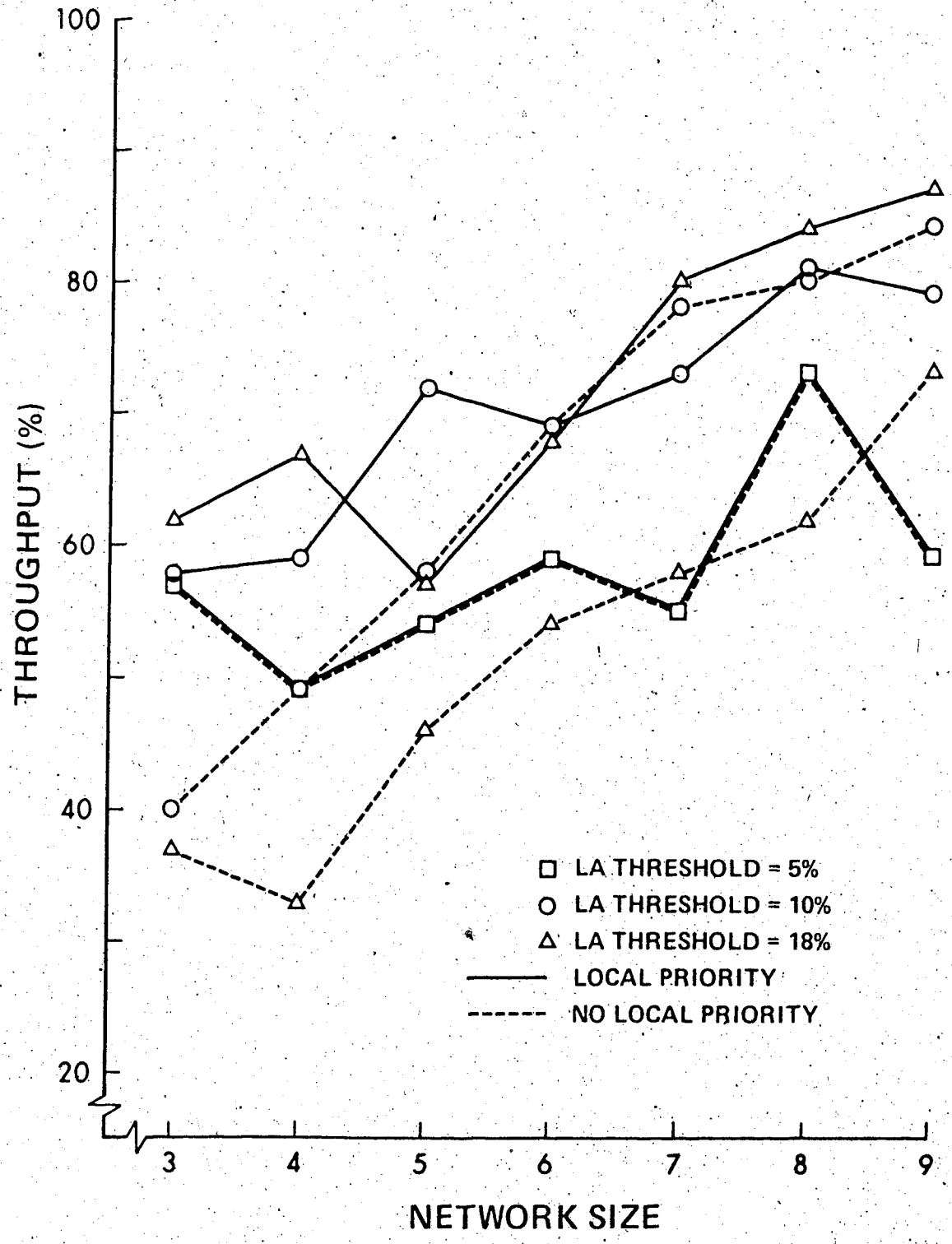

Figure 3a
Throughput by Network Size
(Fragmentation Control Series, First Set)

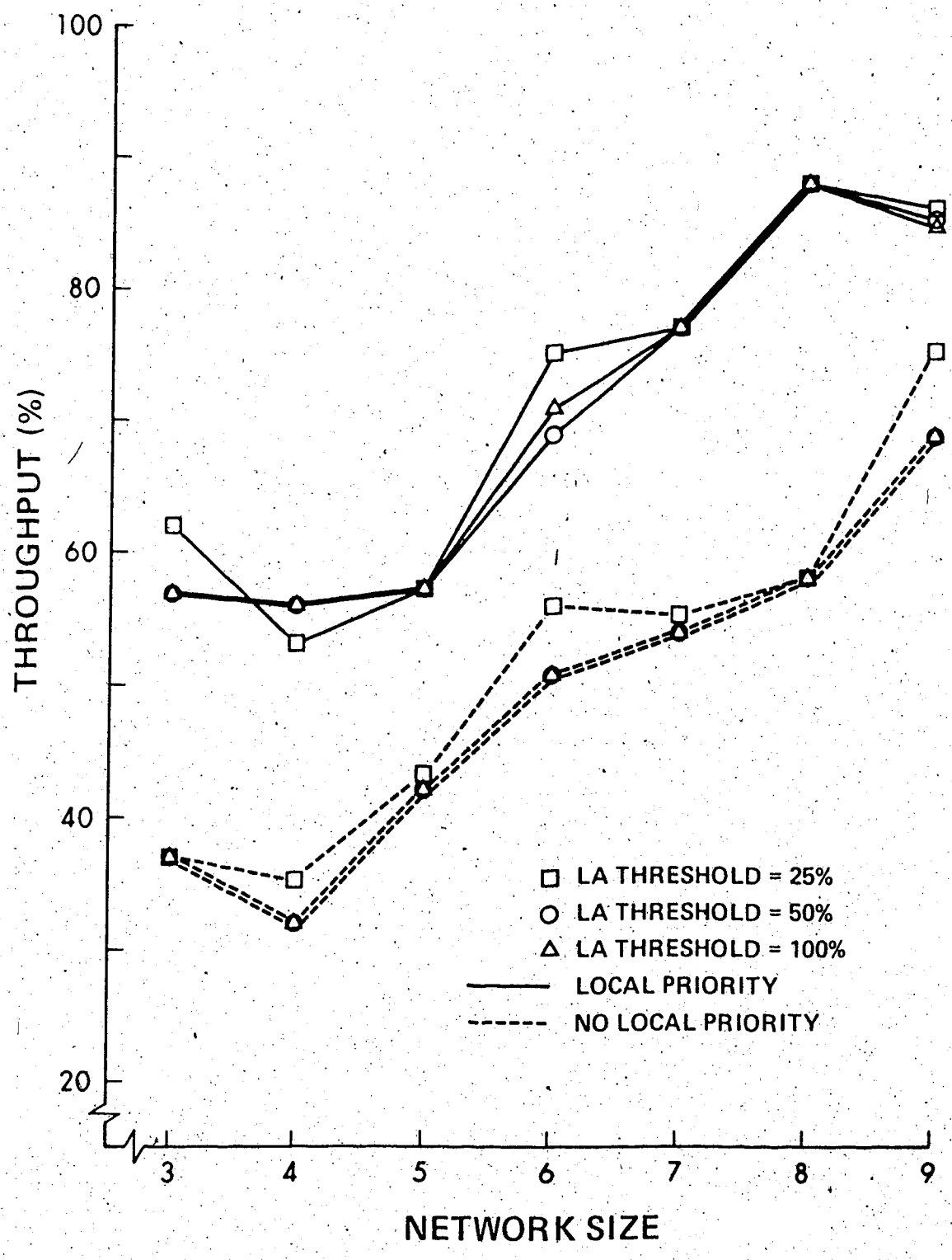

Figure 3b
Throughput by Network Size
(Fragmentation Control Series, First Set)

those for LA thresholds of 25, 50 and 100 percent. (For the sake of legibility the results were divided into two graphs).

Almost without exception experiments using the LP strategy (solid lines) show greater throughput than those in which this strategy was not included (dashed lines), especially those experiments with LA thresholds of 18 percent and more. A limited acceptance threshold of 5 percent gave identical results with and without the LP strategy. It would seem that this threshold is too small, as experiments which include both the LP strategy and a threshold of 10 and 18 percent yield better throughput than the 5 percent threshold. (The ratio of average memory request of a remote job to memory size at a node in these experiments is 6500 / 64000 $\cong$ 10 percent, so a 5 percent threshold would be too small for many remote jobs). In Figure 3b the values for network throughput with the LP strategy and thresholds of 25, 50 and 100 percent are very similar, which would seem to indicate that no further improvement in throughput can be gained by increasing the threshold beyond 25 percent.

An additional observation is that in experiments with the LP strategy throughput tends to increase with network size, as there are more remote nodes to choose from. Reference to Figure 2 confirms that remote utilization increases with network size in experiments using the LP

strategy.

Average turnaround time for different network sizes is shown in Figures 4a and 4b. In most cases experiments using the LP algorithm produce shorter turnaround times than the corresponding experiments without this strategy. Performance measurements of the latter experiments show longer times when devices could not process jobs due to lack of available memory, and this is likely the cause of longer job turnaround times. In the LP experiments, turnaround tends to be erratic with increasing network size, although a decreasing trend may be postulated. Optimal values for average turnaround occurred in experiments using the LP strategy and a threshold of 5 or 10 percent.

Figures 5a and 5b show average CPU utilization with different network sizes for experiments in this set. An important observation here is that the lines in these graphs are almost identical to those in Figures 3a and 3b respectively. It would appear that CPU utilization is directly proportional to throughput. A decrease in CPU utilization is attributed (in performance statistics produced by the simulation) to lack of available memory for processing jobs, which should (and does) result in decreased throughput.

The second set of experiments in the fragmentation control series has a smaller load factor of 1.0. Limited acceptance thresholds of 10, 18, 25 and 100 percent were

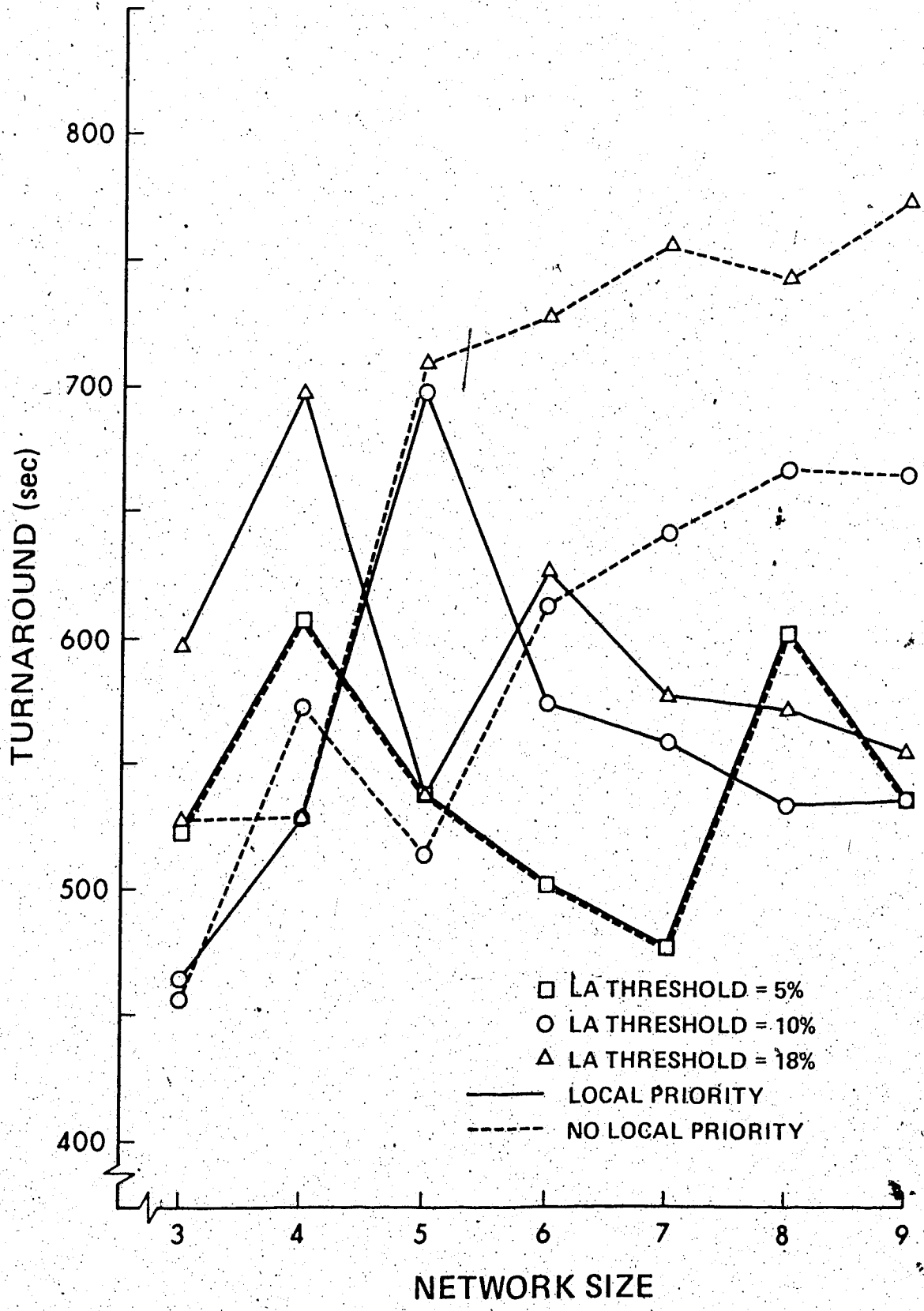

Figure 4a
Turnaround by Network Size
(Fragmentation Control Series, First Set)

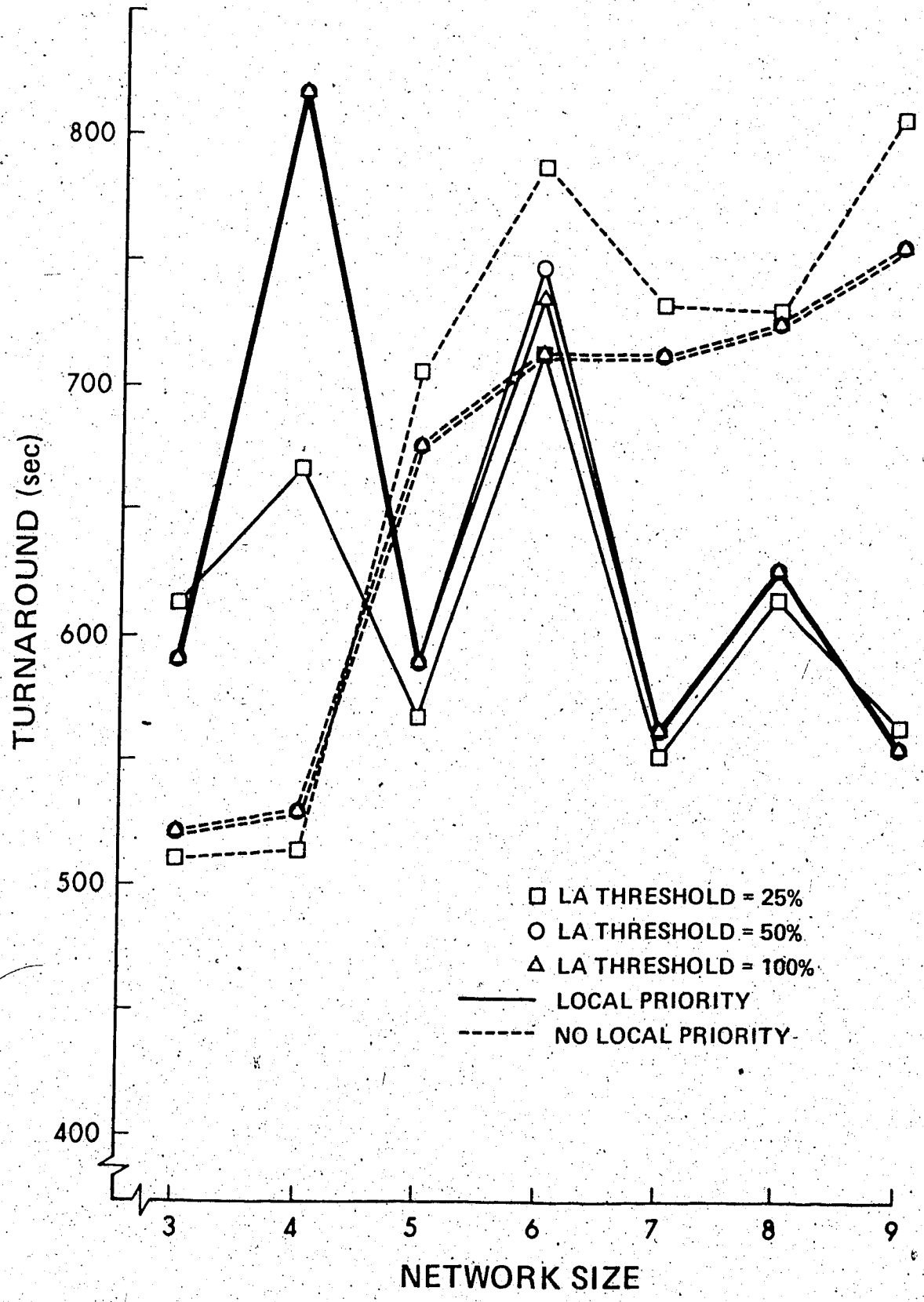

Figure 4b
Turnaround by Network Size
(Fragmentation Control Series, First Set)

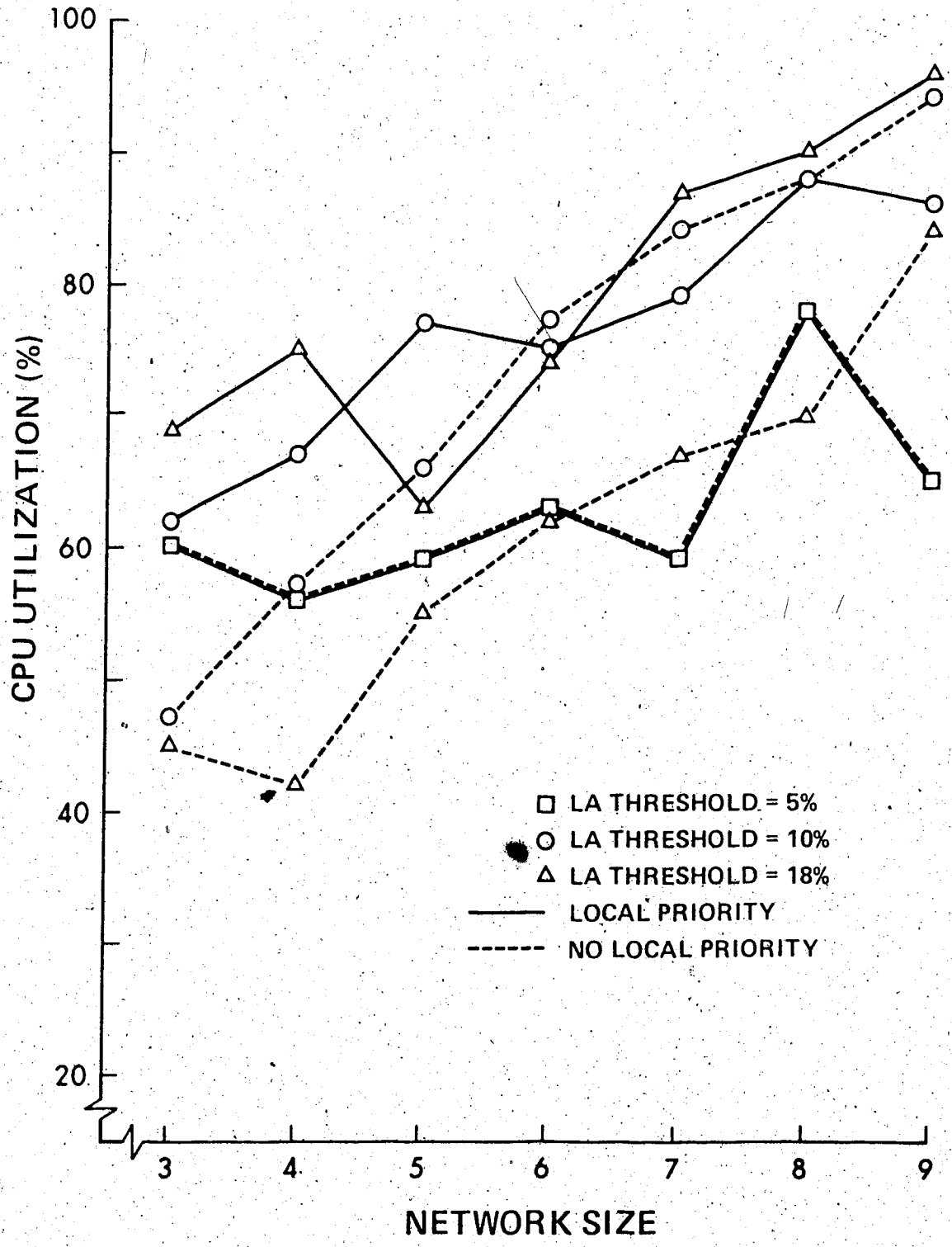

Figure 5a
CPU Utilization by Network Size
(Fragmentation Control Series, First Set)

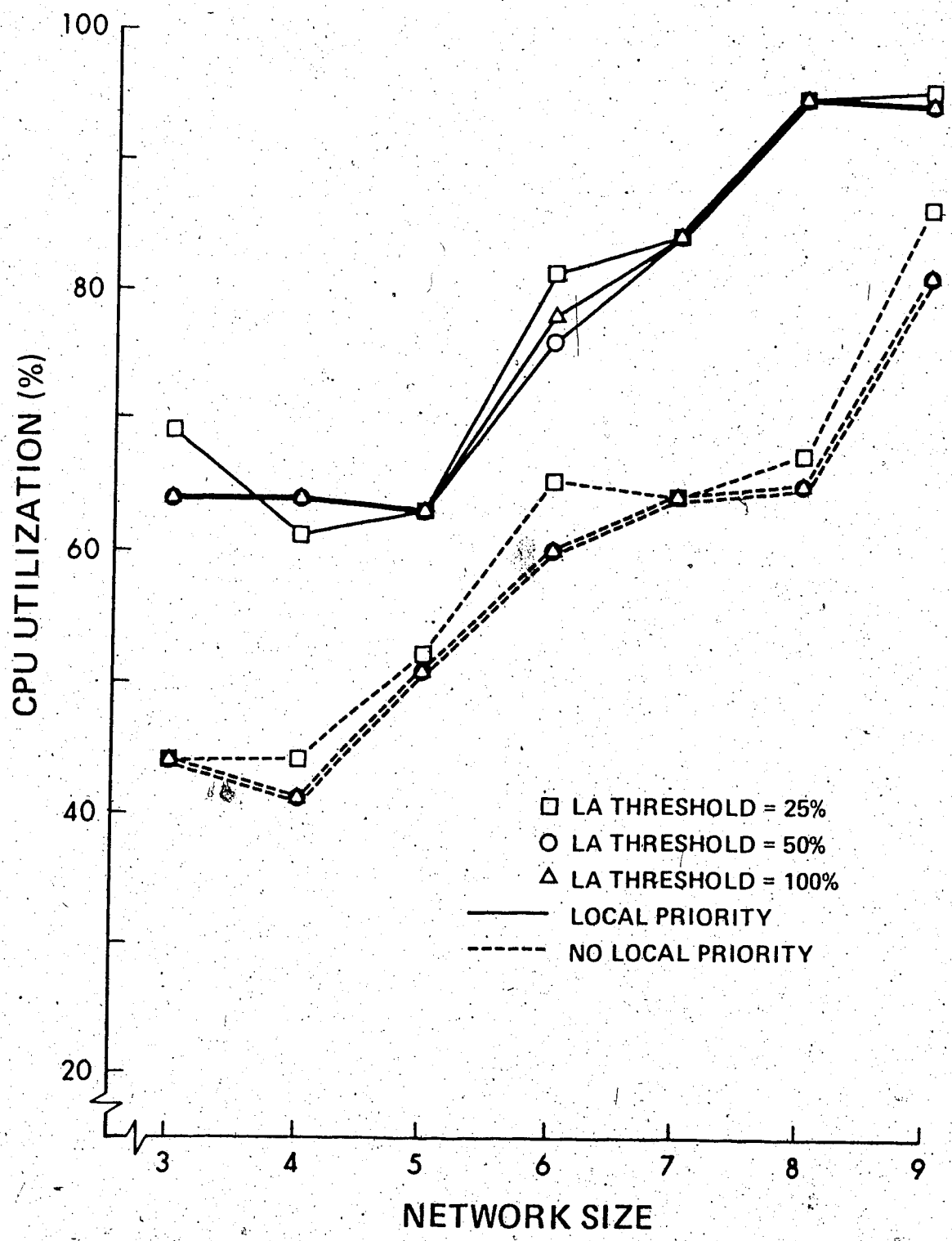

Figure 5b
CPU Utilization by Network Size
(Fragmentation Control Series, First Set)

tested with and without the LP strategy. The results of these experiments are given in Figures 6, 7, 8 and 9.

A large[illegible]se in throughput is observed as remote utilization [illegible] in Figure 6. As throughput drops below 50 percent f[illegible]ion occurs. As in the first set this corres[illegible]s t[illegible]e experiments with the absence of the LP algorit[illegible] Aga[illegible] the optimal throughput values occur with remote u[illegible]diz[illegible]on of about 10 percent, and these are experimen[illegible]ch include the LP strategy.

Figur[illegible] shows the improved throughput of experiments using the L[illegible]strategy over those without, but no significant difference i[illegible]throughput is observed with any of the four thresholds. [illegible] latter observation is likely due to the lighter network load.

Turnaroun[illegible]imes in Figure 8 are much less in the LP experiments t[illegible]n those without LP. The turnaround times of the former group decrease as network size increases, and optimal turnaround values occur with a 10 percent threshold. It is to be noted in this set of experiments that the values for throughput have increased and those for turnaround have decreased from the experiments in the first set, a phenomenon which should intuitively occur with a lighter load.

Figure 9 displays the average CPU utilization by network size for experiments with different thresholds in

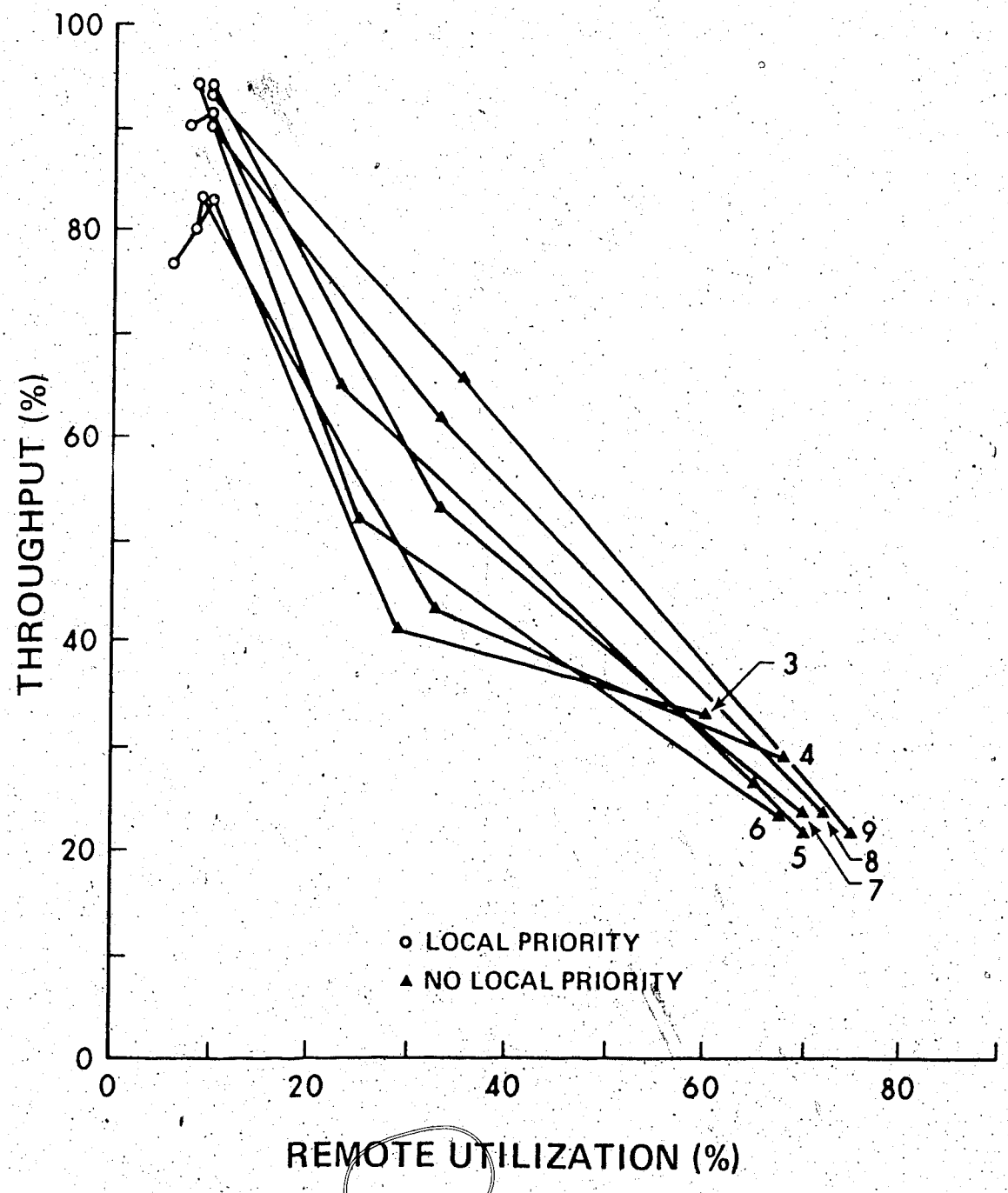

Figure 6
Throughput by Remote Utilization
(Fragmentation Control Series, Second Set)

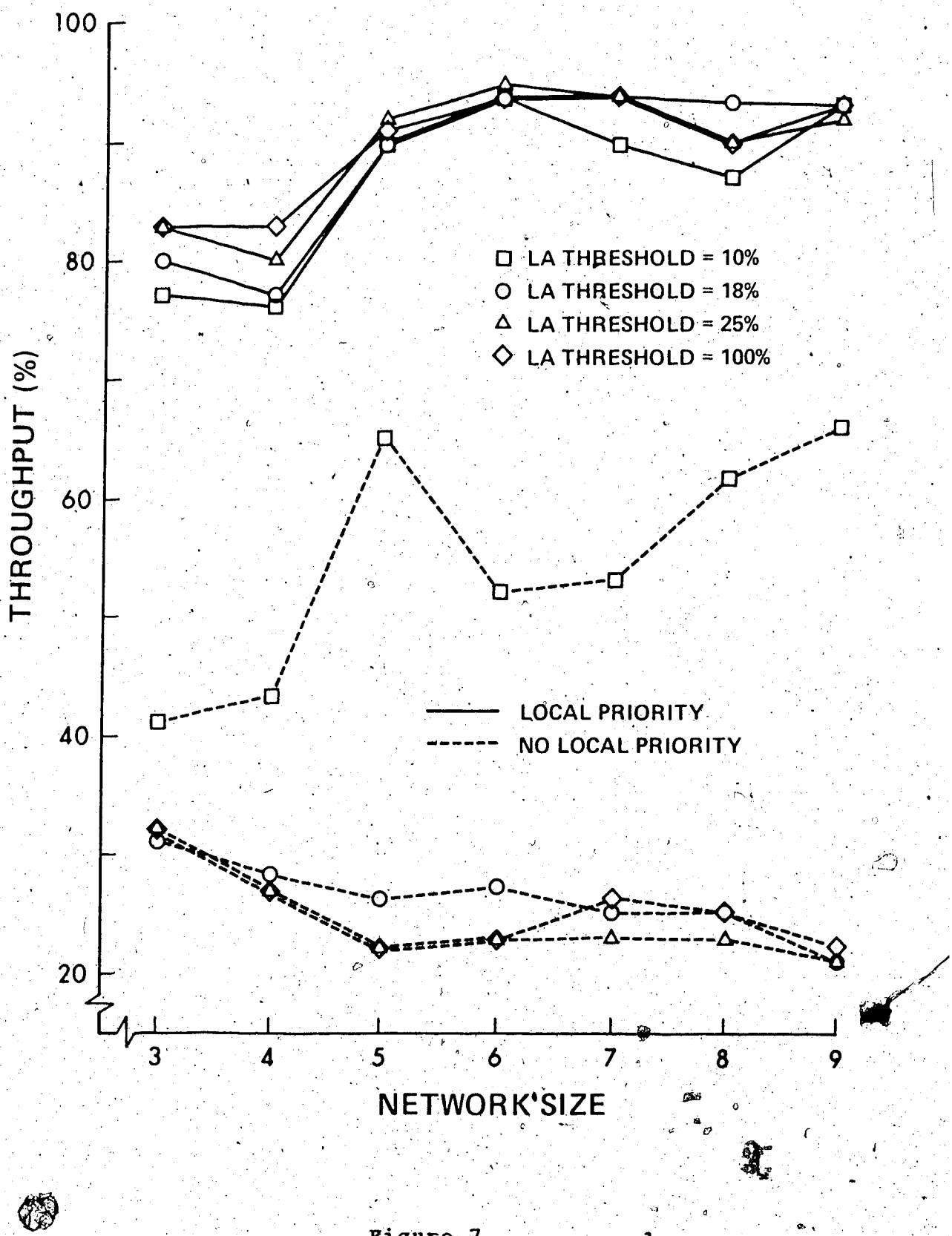

Figure 7
Throughput by Network Size
(Fragmentation Control Series, Second Set)

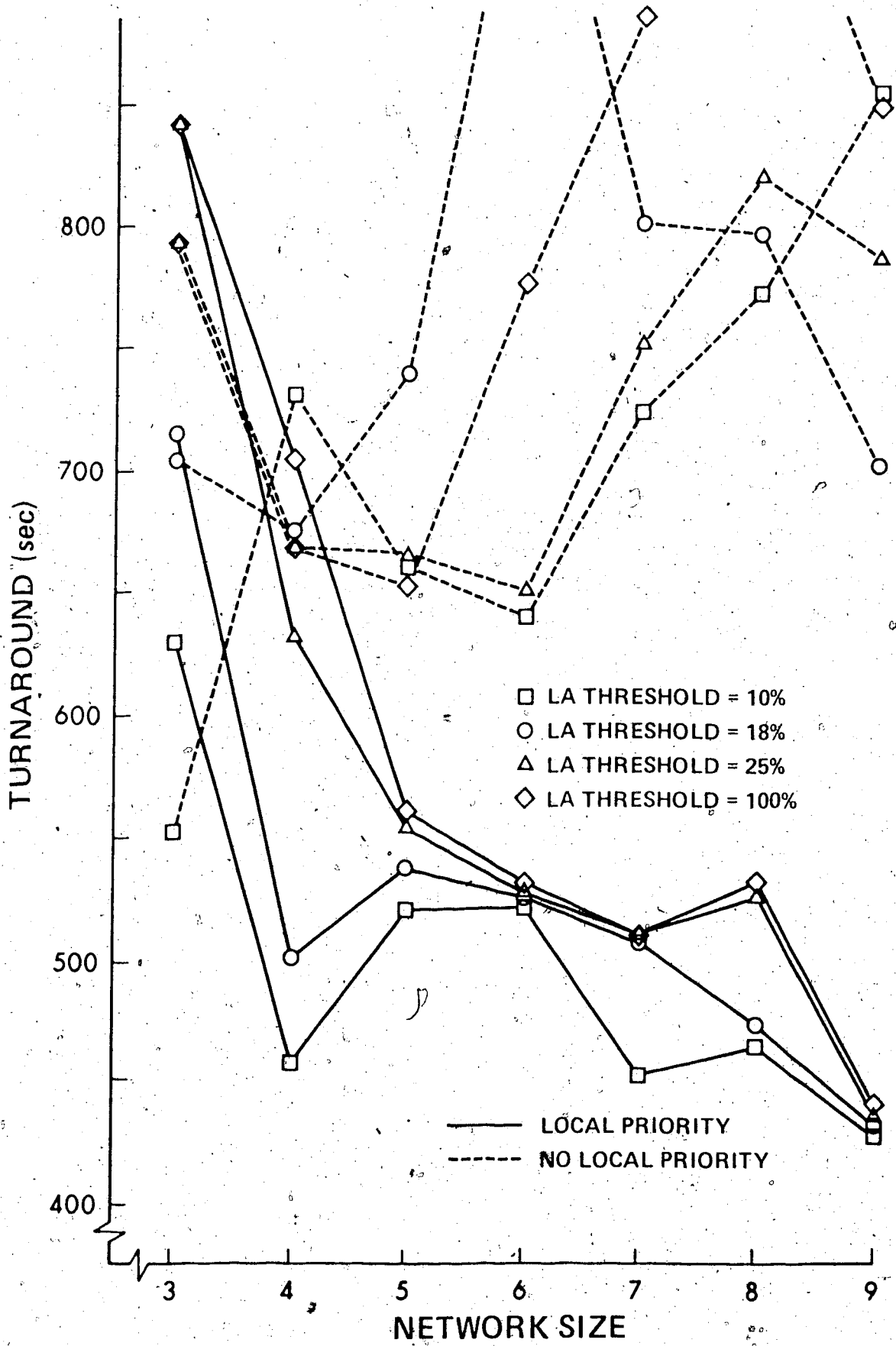

Figure 8
Turnaround by Network Size
(Fragmentation Control Series, Second Set)

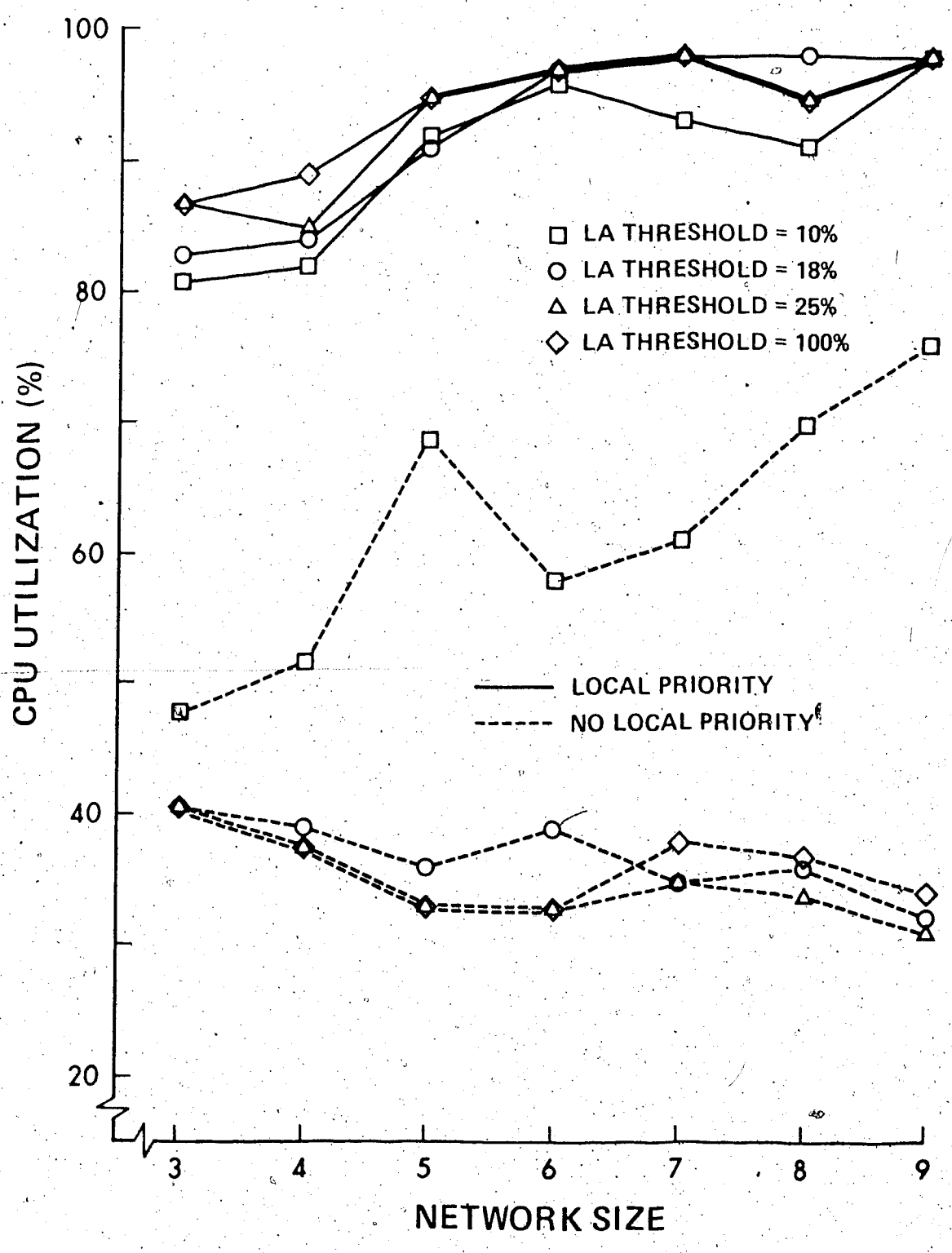

Figure 9
CPU Utilization by Network Size
(Fragmentation Control Series, Second Set)

this set. As was the case in the first set, the graph is almost identical to the graph of throughput by network size (Figure 7).

4.2.2 Comparison of Strategies

The second series of simulation experiments compared the two scheduling strategies introduced in Chapter III - the least transmission cost (LTC) and least memory requirement (LMR) strategies - with each other and with the combination of local priority and limited acceptance strategies which resulted in optimal performance measurements in the first series. Limited acceptance thresholds of 10, 18, 25 and 100 percent were used with each of the three strategies in the first set of experiments in this series. A load factor of 1.09 (the heavier load) was also used.

The three solid lines in Figure 10 show the maximum throughput obtained of the four LA thresholds employed with each of the three strategies. (An explanation of the dashed line is given with the description of the succeeding graph, Figure 11). For all network sizes the greatest throughput values were obtained with the LTC strategy. This strategy gives more priority to remote jobs awaiting retransmission to their input node for output which may explain its superiority to the first-come-first-served strategy. The major failing of the LMR strategy is in the larger amounts

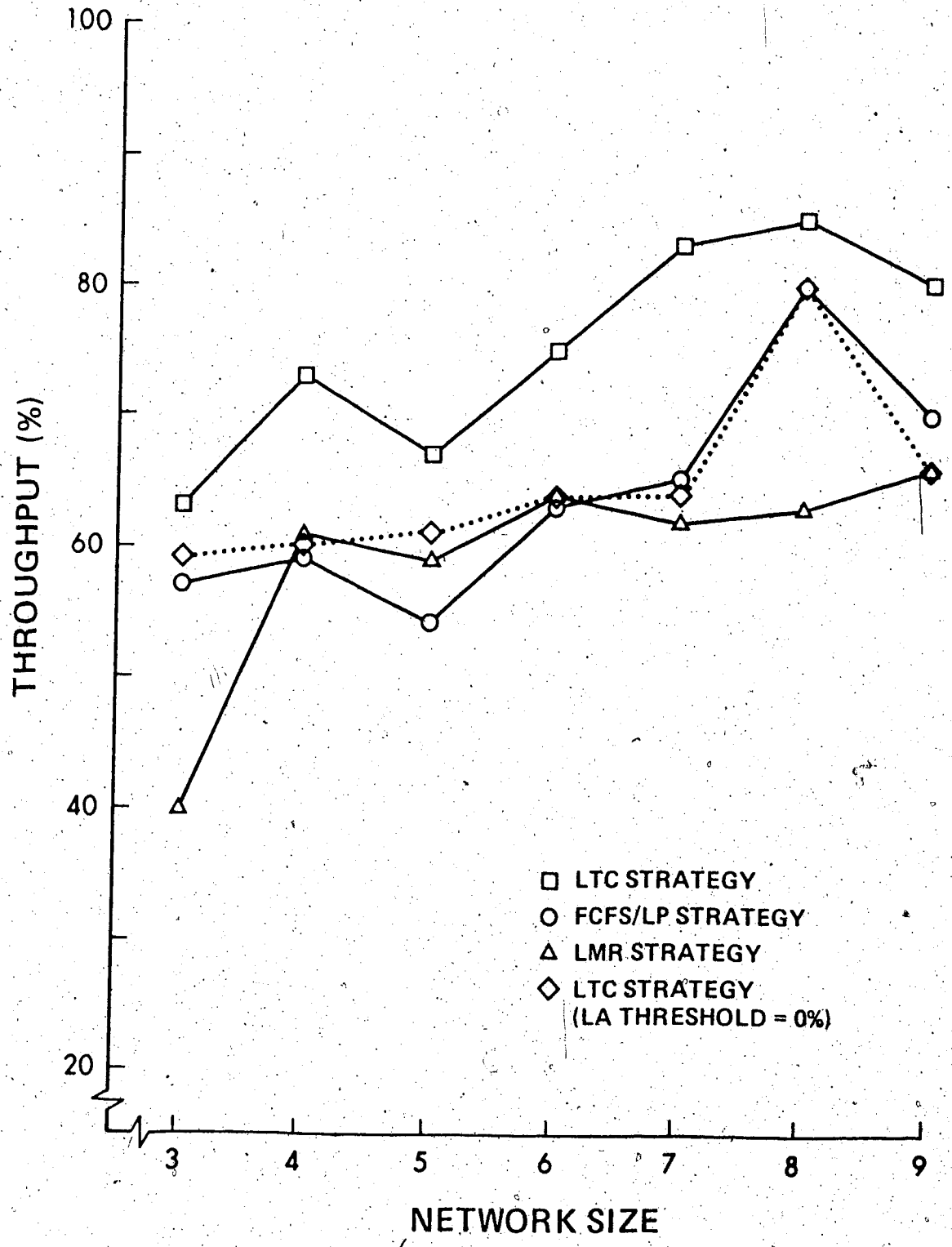

Figure 10
Throughput by Network Size
(Strategy Comparison Series, First Set)

of time in which devices were idle for lack of memory.

	LTC	FCFS/LP	LMR
10%	3	1	4
18%	3	3	2
25%	1	1	1
100%	0	2	0

TABLE I
Limited Acceptance Thresholds yielding
Optimal Throughput Values

Table I indicates the limited acceptance thresholds at which the best throughput occurred for each strategy. It is readily apparent that no single threshold produced the majority of optimal throughput values, although the 10 and 18 percent thresholds showed optimal throughput more often than the 25 and 100 percent thresholds. This result is consistent with a similar observation of Figures 3a and 3b.

A comparison of remote utilization which resulted from each strategy is shown in Figure 11. In all cases remote utilization is less than 20 percent, as each strategy in its own manner gives more priority for execution to local jobs than to remote jobs, as explained in Chapter III. As was the

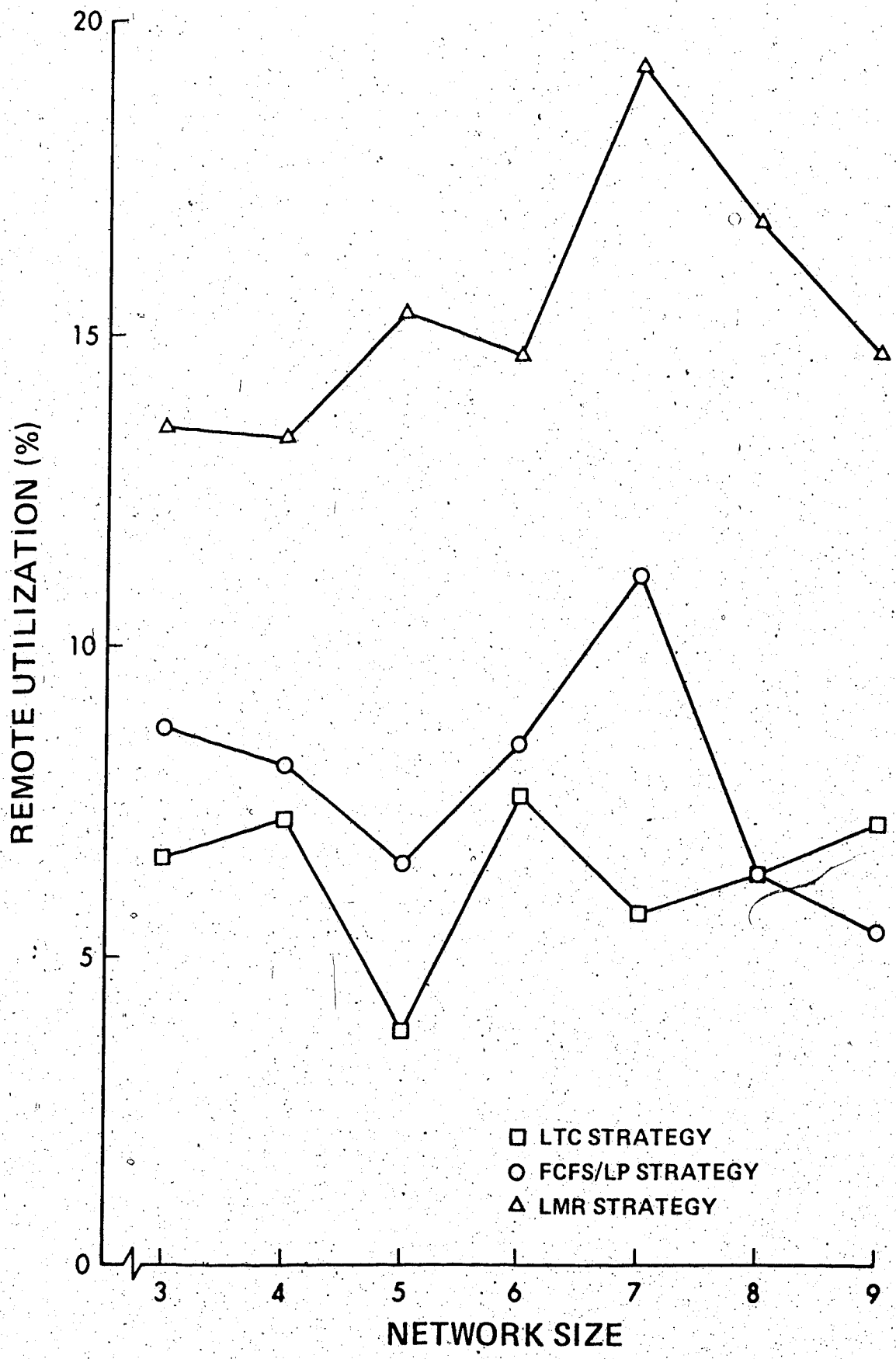

Figure 11
Remote Utilization by Network Size
(Strategy Comparison Series, First Set)

result in the first series, increased remote utilization corresponds to decreased throughput and less remote utilization to higher throughput. In order to test the result of reducing remote utilization to 0 percent, experiments were performed with the LTC strategy and a threshold of 0 percent. The throughput values which resulted are displayed as a dashed line in Figure 10. The decreased throughput indicates that the lower bound on optimal remote utilization is greater than 0 percent i.e. the network can process more jobs than the independent nodes.

Figure 12 shows average turnaround times for each of the three strategies. The LTC strategy which resulted in the greatest throughput values unfortunately also produces the longest turnaround for jobs. Turnaround times for the oldest/local priority strategy are similar, but the LMR strategy gave low turnaround times corresponding to its low throughput values.

Average CPU utilization in Figure 13 exhibits the same property as shown in previous experiments, namely that its graph corresponds with the throughput graph (Figure 10). In all cases optimal throughput and optimal CPU utilization occurred at the same limited acceptance threshold.

As in the first series of experiments the second set in the strategy comparison series was conducted with a load factor of 1.0. The observations of Table I resulted in the selection of 10, 14 and 18 percent LA thresholds. Figures 14

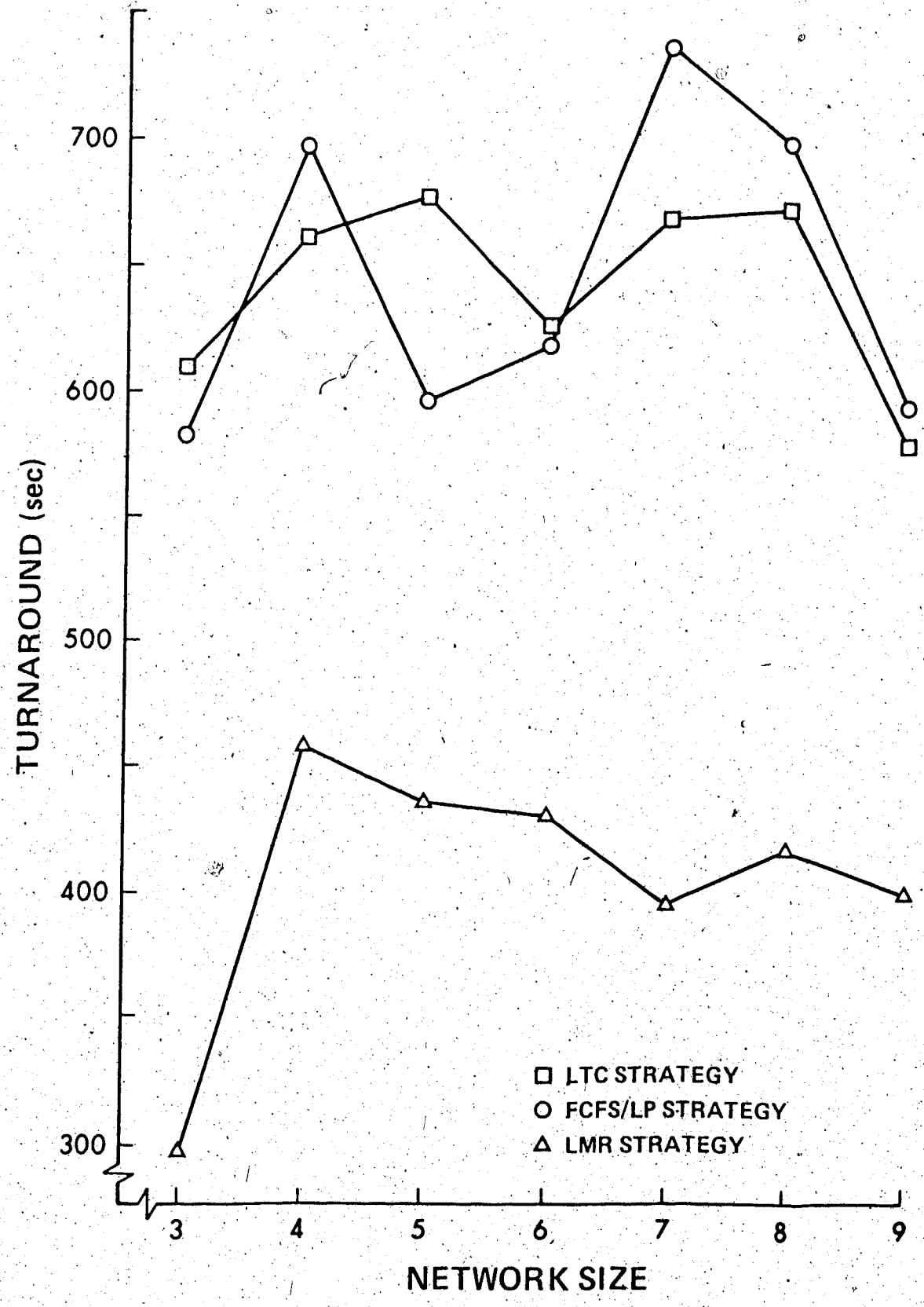

Figure 12
Turnaround by Network Size
(Strategy Comparison Series, First Set)

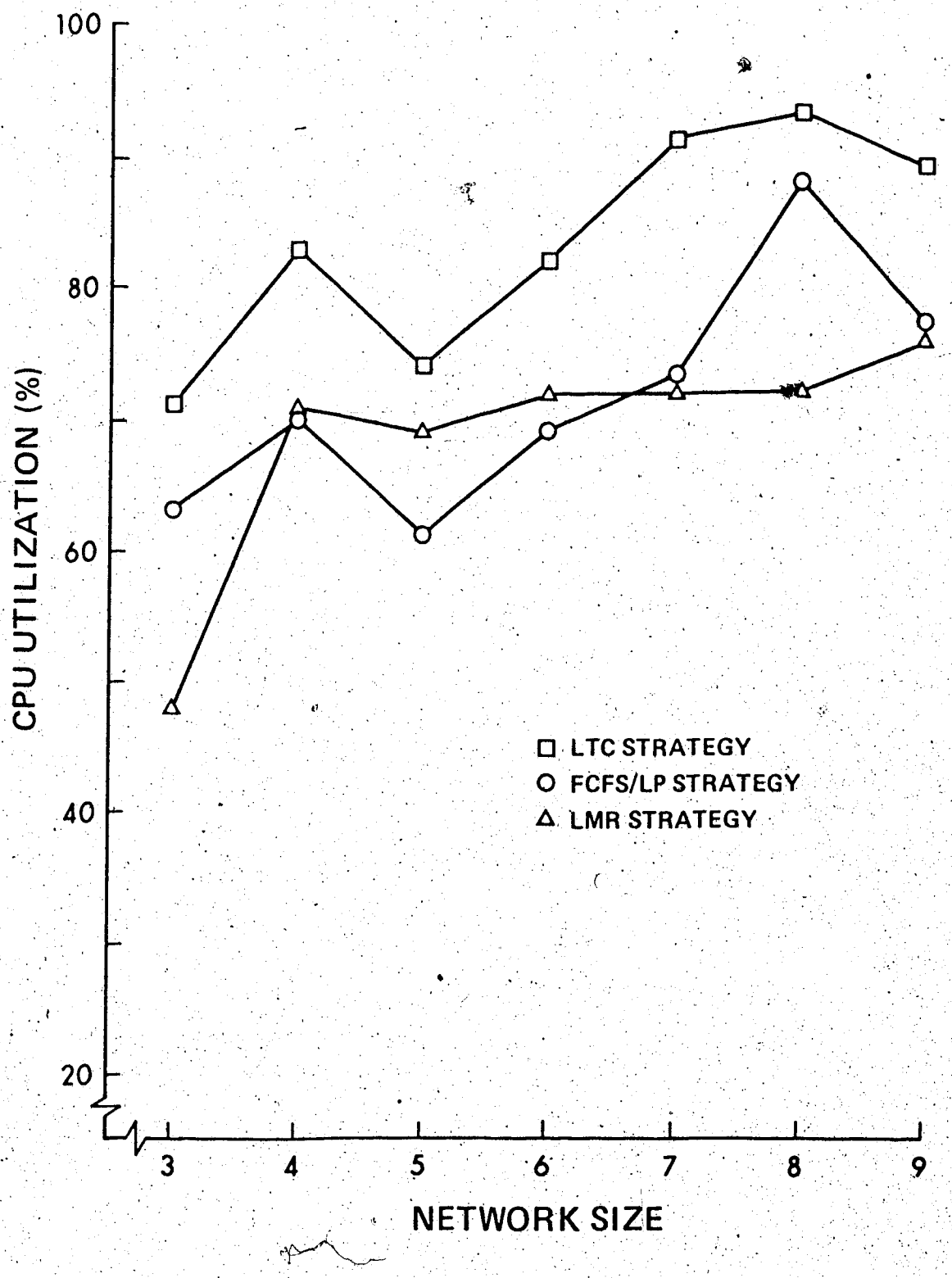

Figure 13
CPU Utilization by Network Size
(Strategy Comparison Series, First Set)

through 17 contain the results of this set.

The values for network throughput shown in Figure 14 are more closely grouped than those in the previous set (Figure 10), as the network load is smaller. The best results still occur with the LTC strategy. Whereas throughput tended to increase with network size in Figure 10, it appears to be constant in this case. Table II indicates that optimal throughput values were most likely to occur at limited acceptance thresholds of 14 and 18 percent.

	LTC	FCFS/LP	LMR
10%	0	0	3
14%	2	4	3
18%	5	3	1

TABLE II
Limited Acceptance Thresholds yielding
Optimal Throughput Values

Contrary to expectation based on results of the previous set the strategy yielding maximum throughput was not the same as that producing maximum remote utilization, as Figure 15 indicates. Although the throughput values for the LTC and FCFS/LP strategies were similar the remote utilization values for the LTC strategy are larger than

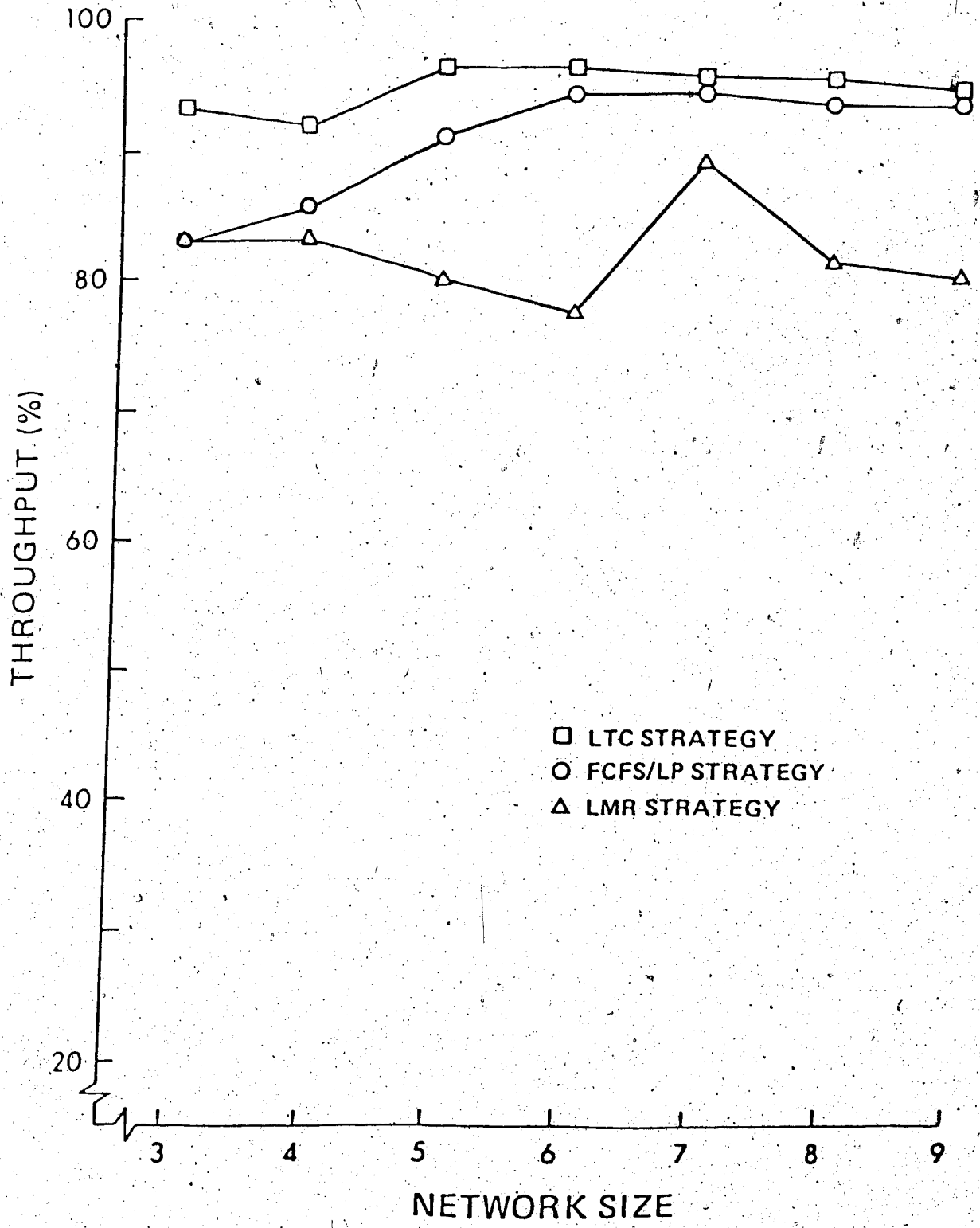

Figure 14
Throughput by Network Size
(Strategy Comparison Series, Second Set)

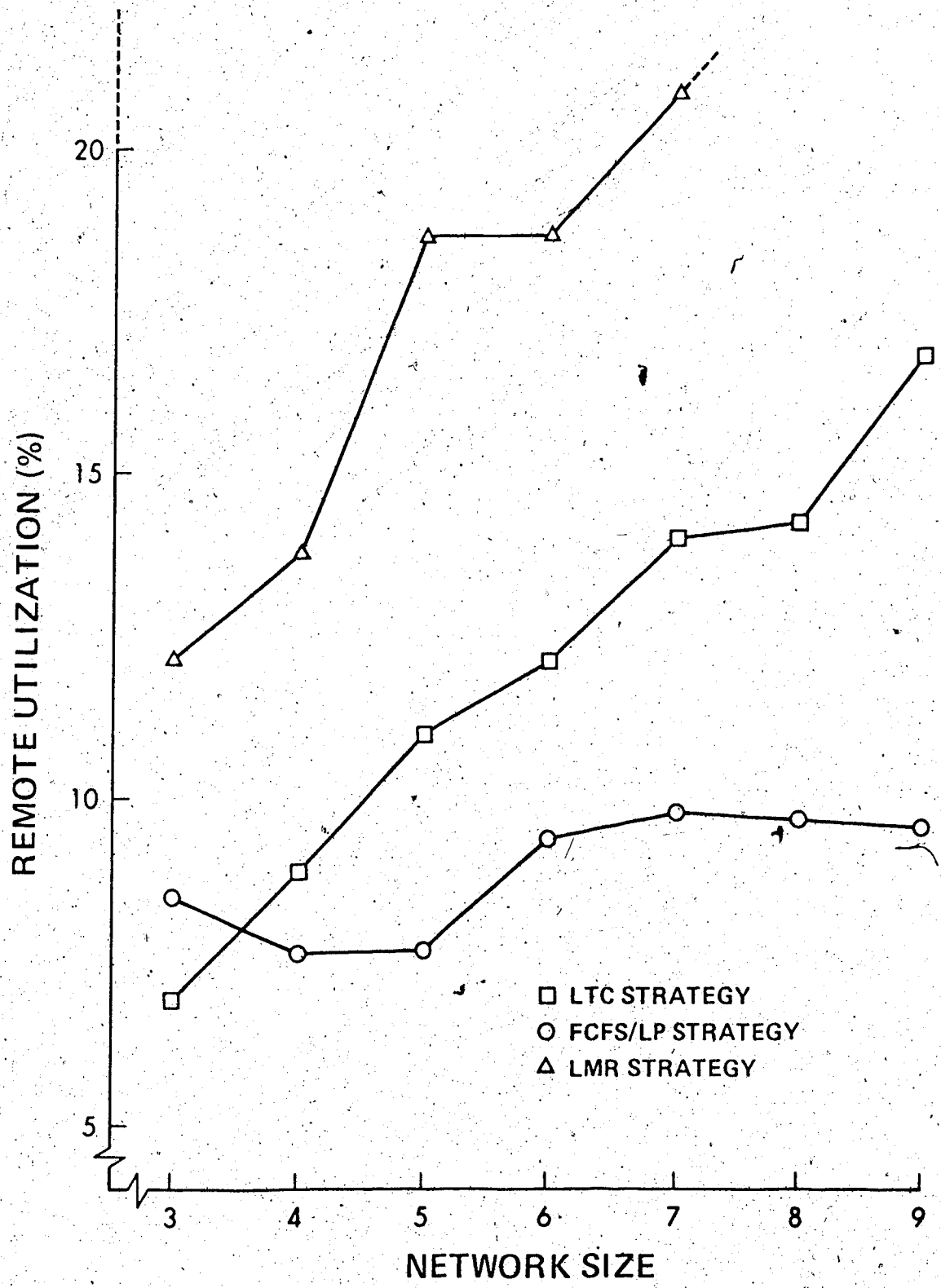

Figure 15
Remote Utilization by Network Size
(Strategy Comparison Series, Second Set)

those for the FCFS/LP strategy.

Figure 16 shows turnaround times for this set. As was the case in the fragmentation control series with a lighter load, turnaround times tend to decrease with increasing network size. The LTC strategy generally produced the lowest values for turnaround times.

Figure 17 continues the trend of maximum CPU utilization corresponding to maximum throughput. The LA threshold wherein both occurred is in all cases the same one.

The next section summarizes the results of the two series of experiments. Significant factors which may have caused the results are discussed.

4.3 Summary

The principal observation apparent in all experiments is that fragmentation results in decreased throughput and CPU utilization and increased turnaround times. The experiments of the first series which used the first-come-first-served strategy without local priority are an example in point. In these experiments a large number of remote jobs were occupying the memory at other nodes, either while awaiting retransmission for output or while executing. (The simulation is constructed so that remote execution always requires at least as much memory as local execution.) The situation is worsened by the possibility of a form of deadlock occurring, in which two or more jobs require memory

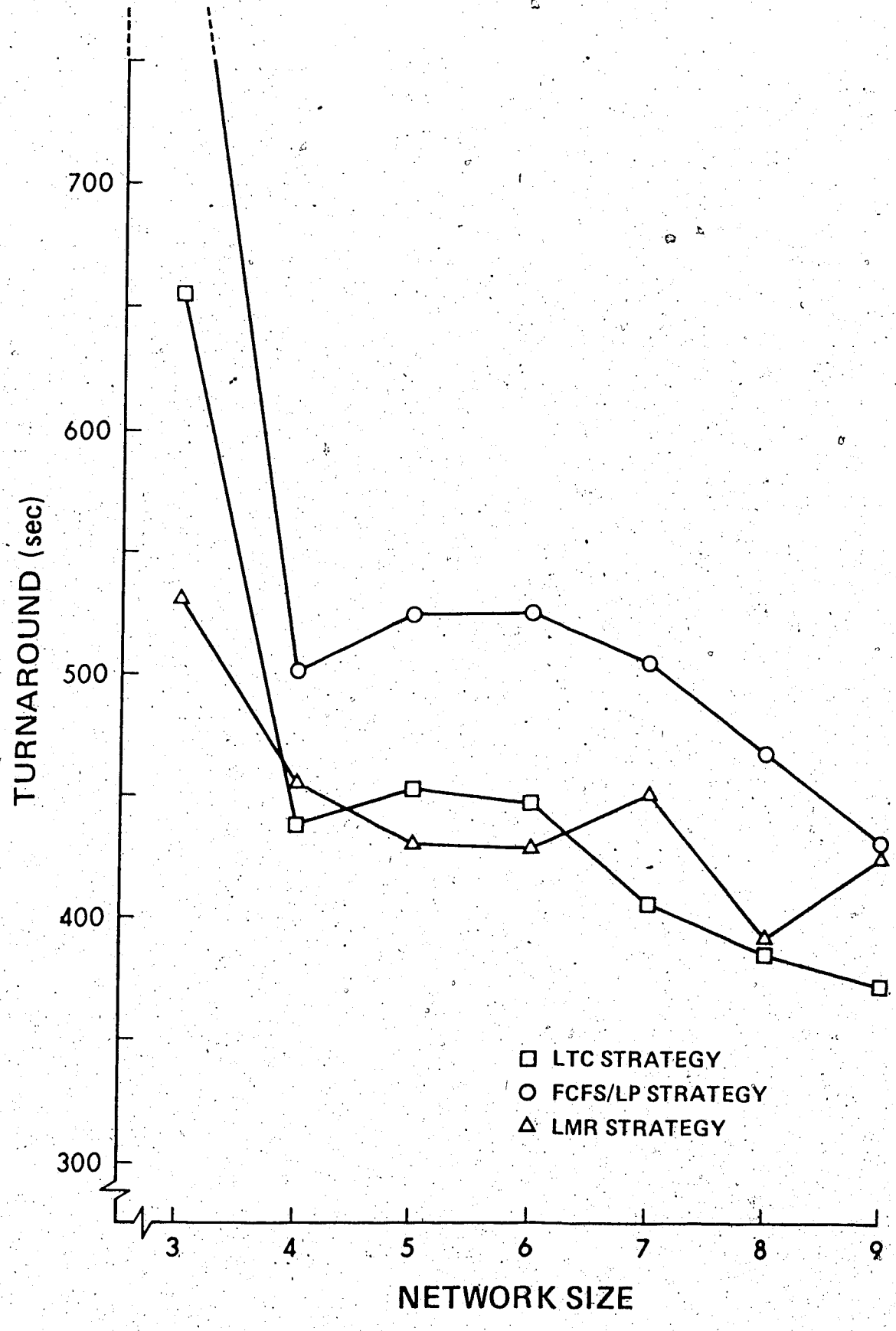

Figure 16
Turnaround by Network Size
(Strategy Comparison Series, Second Set)

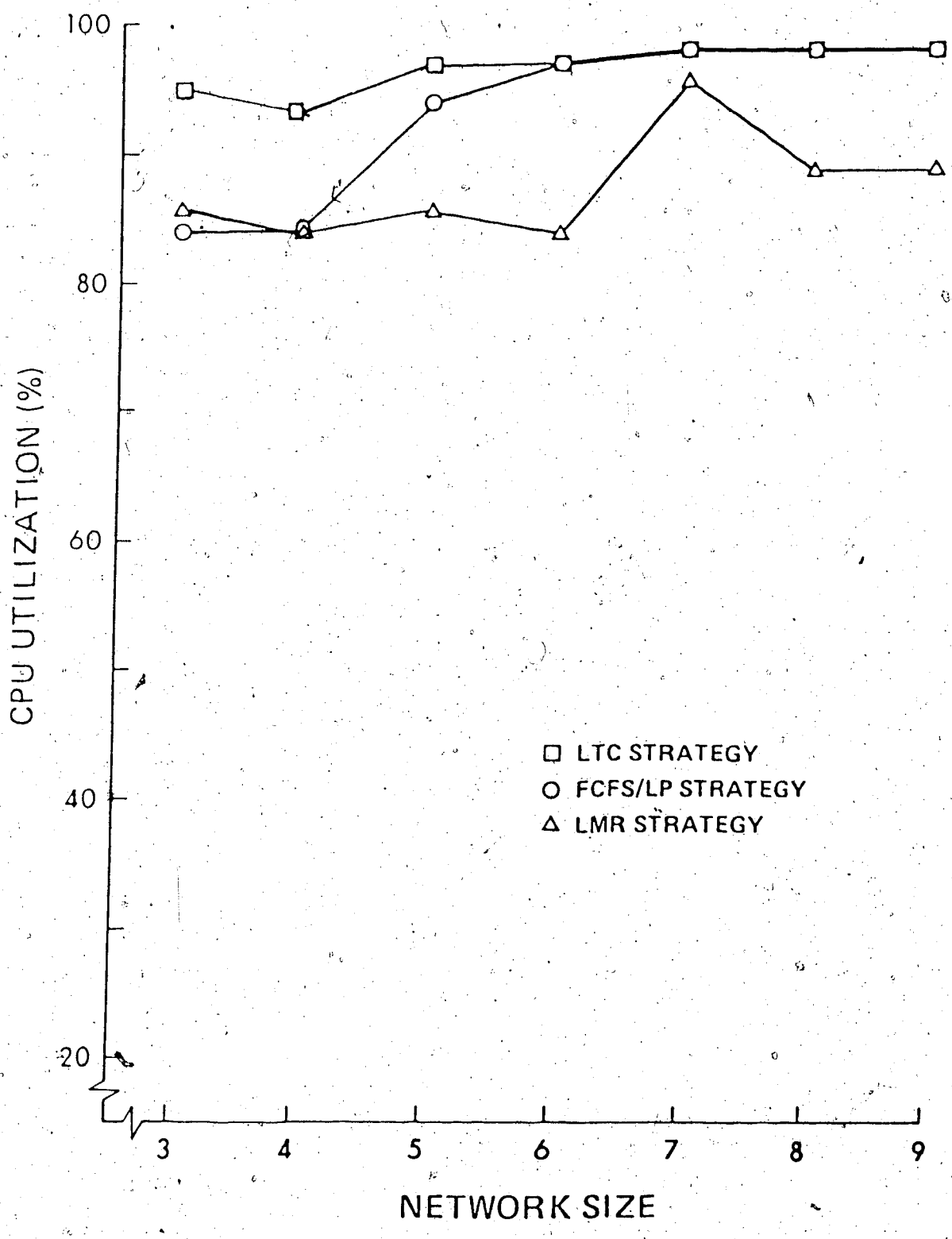

Figure 17
CPU Utilization by Network Size
(Strategy Comparison Series, Second Set)

at the others' nodes. It is clear that fragmentation must be controlled or an overuse of memory by remote jobs will result.

The inclusion of the local priority strategy effectively controls fragmentation, especially with the addition of some limited acceptance threshold. These two strategies together produce better results than either alone. A limited acceptance threshold of 10 to 18 percent of the memory size at a node is optimal in most cases. Larger thresholds did not yield significantly better performance statistics. The discovery of an ideal limited acceptance threshold may be affected by some or all of the following parameters: network size, network load, scheduling strategy used, ratio between average memory request and memory size of a node, and the average number of jobs executing at the same time (assuming a multiprogramming environment).

The least transmission cost strategy gave optimal results in the strategy comparison series of experiments. Despite its property of having no priority based on the age of jobs, turnaround times with this strategy were not significantly increased over those of other strategies under a lighter load, although they were longer than those of other strategies under a heavy load. This would seem to indicate that the LTC strategy would perform better under conditions of heavier loads if some age factor were included in its priority ranking scheme.

Memory size and availability were crucial factors in the experiments. As memory in the simulation was a finite resource, the network tended to have poorer throughput and jobs to have longer turnaround times when no memory was free. Device utilization also decreased. As this restriction of the simulation influenced some of the results, further tests would be necessary to determine the degree of dependency of the results obtained on this factor.

Conclusions which may be drawn from the results obtained in this chapter form the basis of the next chapter. Suggestions for further research in this area are also proposed.

CHAPTER V

CONCLUSIONS

In the previous chapter two series of experiments of load sharing networks were conducted with the aid of a simulation program, and their results were summarized. This chapter contains a brief review of the results and their validity as output of a computer simulation. The chapter concludes with some potentially profitable avenues of further research.

5.1 Review of Results

One important outcome is due to each of the experimental series. The major result of the first series has been the demonstration of a retrogression in network performance parameters caused by fragmentation. It has been demonstrated that fragmentation can be adequately controlled by a number of methods. Three proposed scheduling strategies for load sharing networks were tested in the second series of experiments, which examined their properties and their prospects. A scheduling algorithm based upon least cost of transmission for remote jobs was found to produce optimal values of performance statistics.

Consideration should be given to the fact that the above are results of a limited number of tests with particular input parameters. It must be assumed that the

simplifications inherent in the simulation (for example, the finite memory size, absence of multiprogramming and limitations on network configuration) are not so restrictive as to be completely responsible for the results obtained.

The value of the results obtained from the experiments which were conducted is their usefulness as guidelines for the continuation of research into strategies for load sharing. Additional experiments will yield more insight and may prove to verify these results.

5.2 Directions for Further Research

Extensions to the theory described and the experiments conducted are of three types. Further experimentation may be done with the simulation as is, the simulation and scheduling strategies may be extended, or consideration can be given to the implementation of successful strategies.

As a further test of the results produced, input parameters to the simulation can be altered. In this manner asymmetrical loads and varying job mixes (e.g. I/O- and CPU-bound jobs) can be simulated. The limited acceptance strategy requires examination under simulated networks with nodes of unequal memory sizes as an aid in controlling memory allocation, a factor which contributed to some of the results.

If some of the simplifications used in the simulation were removed, it would be seen if the results apply to a

more general case. The complete graph and fixed memory size restrictions, and lack of multiprogramming are prime targets for this improvement.

A potential extension to the scheduling strategies is the inclusion of user-specified priorities based on execution time, desired turnaround time or other factors, to be used in conjunction with the existing strategies. Turnaround times of those strategies which do not include an aging factor would be increased by the addition of an algorithm to elevate a job's priority as a function of its age, in co-operation with the original scheduling scheme. An "ideal" strategy for a specific network would probably be tailored to the network's characteristics and adaptable to the changing network load. Limited acceptance thresholds modifiable as a function of the state of the network would be an example of the latter property.

Preceding actual implementation of resource sharing networks in the future, will be the development of specialized operating system software. A study of the requirements for implementation of scheduling strategies in distributed networks would be a fruitful extension. No single host would have authority, and thus the hosts would co-operate in the exchange of resources and jobs. As numerous deadlock situations are possible, techniques used in the scheduling of processes [B1] will be of use here.

The increasing viability of computer networks continues

to be the motivation for much research in networking. By sharing resources and workloads networks are providing valuable services and justifying their existence. For this reason, the extension and development of strategies for job scheduling in resource and load sharing networks will have considerable benefits.

BIBLIOGRAPHY

[A1] Abramson, N., Kuo, F.F. (Eds.) Computer-Communication Networks; Prentice-Hall, New Jersey, 1973.

[A2] Akkoyunlu, E., Bernstein, A., Schantz, R. "Interprocess Communication Facilities for Network Operating Systems"; Computer 7(6), (June 1974), pp. 46-55.

[B1] Brinch Hansen, P. Operating System Principles; Prentice-Hall, New Jersey, 1973.

[BW1] Bowdon, E.K. Sr., Barr, W.J. "Cost Effective Priority Assignment in Network Computers"; Proceedings of FJCC 41, (1972), pp. 755-763.

[BW2] Bowdon, E.K. Sr., Mamrak, S.A., Salz, F.R. "Simulation - A tool for performance evaluation in network computers"; Proceedings of NCC 42, (1973), pp. 121-131.

[C1] Chambers, J.A., Poore, R.V. "Computer Networks in Higher Education: Socio-Economic-Political Factors"; Communications of the ACM 18(4), (April 1975), pp. 193-199.

[C2] Chou, W. "Computer Communication Networks - The parts make up the whole"; Proceedings of NCC 44, (1975), pp. 119-128.

[C3] Coffman, E.G. Jr., Kleinrock, L. "Computer scheduling methods and their countermeasures"; Proceedings of SJCC 32, (1968), pp. 11-21.

[D1] Davis, R.M. "Computing Networks: A Powerful National Force"; Computer 6(4), (April 1973), pp. 14-18.

[F1] Fredericksen, D.H. "Describing data in computer networks"; IBM Systems Journal 12(3), (1973), pp. 257-282.

[FR1] Farber, D.J. "Networks: An Introduction"; Datamation 18(4), (April 1972), pp. 36-39.

[FR2] Farber, D.J., Larson, K.C. "The Structure of a Distributed Computing System - Software"; Proceedings - Symposium on Computer Communications Networks and Teletraffic, April 1972, pp. 539-545.

[FR3] Farber, D.J., Larson, K.C. "The System Architecture of the Distributed Computer System - the Communications System"; Proceedings - Symposium on Computer Communications Networks and Teletraffic, April 1972, pp. 21-27.

[FR4] Farber, D.J., Heinrich, F.R. "The Structure of a Distributed Computer System - The Distributed File System"; Computer Communication - Impacts and Implications, Proceedings - First International Conference on Computer Communication 1972, pp. 364-370.

[H1] Helander, G. Resource Sharing in Computer Networks; MSc. Thesis, Department of Computing Science, Faculty of Graduate Studies and Research, The University of Alberta, Fall 1974.

[K1] Kahn, R.E. "Resource Sharing Computer Communication Networks"; Proceedings of IEEE 60(11), (November 1972), pp. 1397-1407.

[KL1] Kleinrock, L. "Analytic and Simulation Methods in Computer Network Design"; Proceedings of SJCC 36, (1970), pp. 569-579.

[KL2] Kleinrock, L., Naylor, W.E. "On measured behaviour of the ARPA network"; Proceedings of NCC 43, (1974) pp. 767-780.

[KL3] Kleinrock, L., "Resource Allocation in Computer Systems and Computer-Communication Networks"; Proceedings of IFIP Congress 1974, pp. 11-18.

[M1] Meier, R.C., Newell, W.T., Pazer, H.L. Simulation in Business and Economics; Prentice-Hall, New Jersey, 1969.

[M2] Mendicino, S.F. "Octopus: The Lawrence Radiation Laboratory Network" in Computer Networks; R. Rustin (ed.), Prentice-Hall, New Jersey, 1972, pp. 95-110.

[M3] McKenzie, A.M. "Some computer network interconnection issues"; Proceedings of NCC 43, (1974), pp. 857-859.

[P1] Pyke, T.N. Jr., Blanc, R.P. "Computer Networking Technology - A State-of-the-Art Review"; Computer 6(8), (August 1973), pp. 13-19.

[R1] Roberts, L.G., Wessler, B.D. "Computer Network Development to Achieve Resource Sharing"; Proceedings of SJCC 36, (1970), pp. 543-549.

[R2] Rubin, D., Deo, N. "An Additive Pseudorandom Number Generator with Semi-Infinite Sequence Length"; Proceedings - Second Hawaii International Conference on System Science 1969, pp. 467-470.

[S1] Saaty, T.L. Elements of Queuing Theory; McGraw-Hill, New York, 1961.

[T1] Thomas, R.H. "A Resource Sharing Executive for the ARPANET"; Proceedings of NCC 42, (1973), pp. 155-163.

[W1] Walden, D.C. "A System for Interprocess Communication in A Resource Sharing Computer Network"; Communications of the ACM 15(4), (April 1972), pp. 221-230.

[W2] Weis; A.H. "Distributed Network Activity at IBM" in Computer Networks, R. Rustin (ed.), Prentice-Hall, New Jersey, 1972, pp. 1-25.

APPENDIX

A COMPUTER NETWORK SIMULATION

I. Description of the Simulation

A computer network is represented in the simulation as a complete graph, so that routing problems need not be considered. Each node consists of the following resources to be allocated to jobs:

- a CPU
- a batch input device (card reader)
- a batch output device (line printer)
- a fixed amount of memory (which represents all storage media)

Batch jobs only are processed by the network, and multiprogramming is not included; that is, only one job can execute at any time and preemption is not permitted. A job is viewed as a series of requests for the use of resources for a specified length of time. The resources requested by jobs are:

- an input device
- a portion of memory for input
- a CPU
- an additional amount of memory for execution
- an output device (at the same node as the input device)
- a portion of memory for output

These resource requests were given as values of random variables from the uniform, exponential and normal distributions (with distribution parameters specified as input to the simulation). Interarrival times of jobs at each node are similarly specified. Job streams generated are

identical for every simulation run with the same network size.

The processing of a job is divided into input, execution and output phases. Resources needed for succeeding phases are sought at the completion of the preceding phase. Jobs may be executed at their input node (local execution) or transmitted to a remote node for execution and then retransmitted to their input node for output (remote execution). At each node there are four queues of jobs, which share the node's memory with the currently executing job. These queues are:

1. Jobs awaiting input (about to be input to the card reader)
2. Jobs awaiting execution
3. Jobs awaiting transmission for output at other nodes (i.e. jobs which were executed remotely at this node)
4. Jobs awaiting output at this node

Figure 18 illustrates the structure of the simulation. The activity of the network is simulated for a specified length of time, this time span being divided into a number of equal-length sampling intervals. The following are the events which occur in each interval:

1. Jobs with arrival times in this interval are generated at each node and appended to the input queues.

2. The next event to be completed in this interval (if such an event exists) is found.

3. If it is a job which has finished input the following are done:
   - the job is assigned to an idle CPU or appended to the execute queue at its local node if no CPU is free

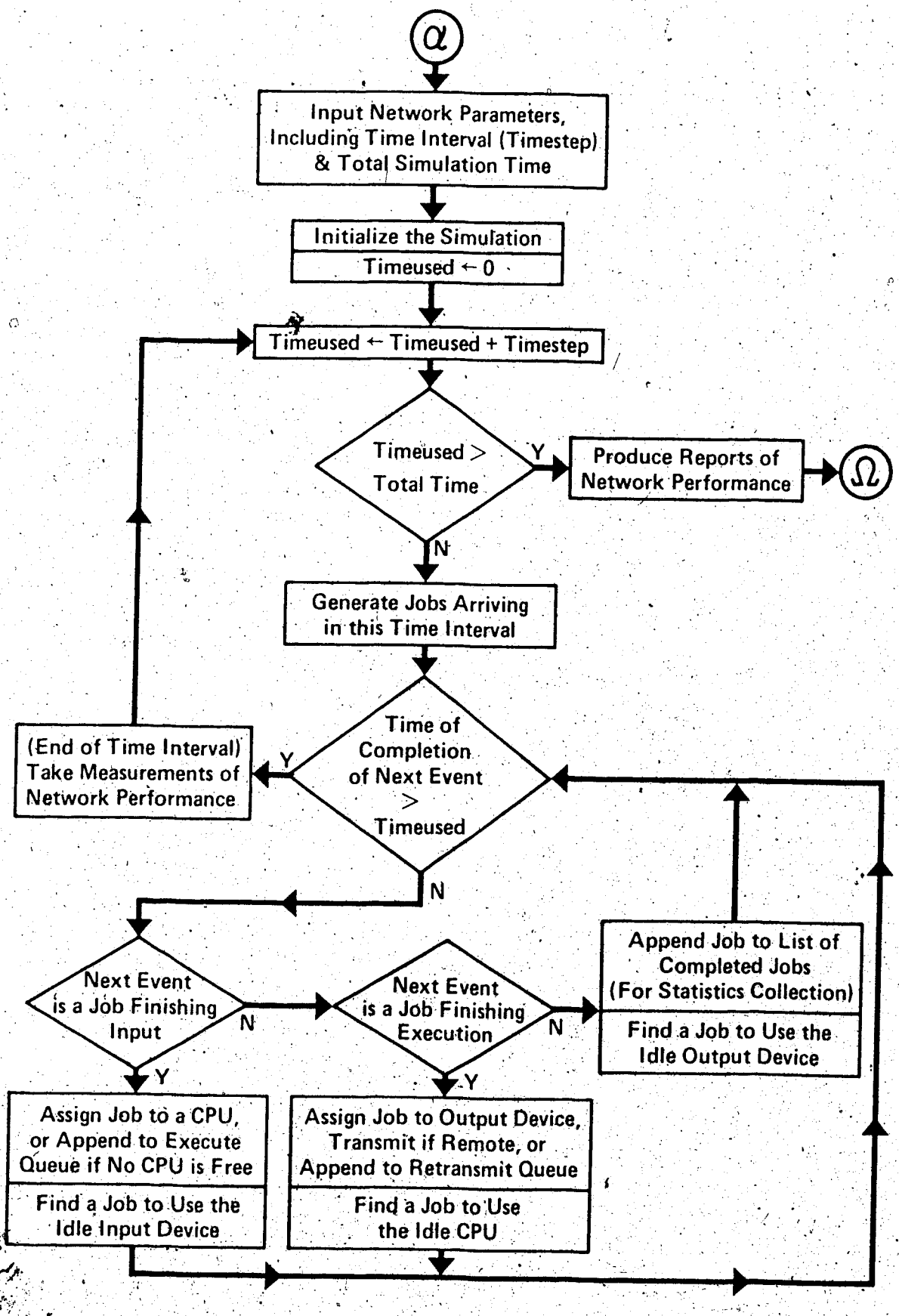

Figure 18
A Flowchart Outline of the Simulation

- the job at the head of the input queue (if there is one) is assigned to the now-idle input device

4. If it is a job which has finished execution, the following are done:
   - the job is assigned to the output device at its input node if the device is free
   - if the job is remote it is transmitted and joins the output queue at the output node if memory is available
   - otherwise a remote job joins the queue at its CPU node of jobs awaiting transmission for output at other nodes
   - ▫ a job is selected to use the now-idle CPU

5. If it is a job which has finished output, the following are done:
   - the job is appended to the queue of completed jobs of which statistics are taken
   - ▫ a job is selected to use the now-idle output device

6. When the end-processing time of the next job to be completed occurs after the end of the time interval, measurements of the network's performance are taken. The length of one time interval is added to the accumulated time, and the process is repeated for the new time interval.

(The symbol ▫ denotes that the current scheduling strategy or strategies are used to select a job or device.)

When the accumulated time exceeds the specified duration of the simulation, reports of statistics of job throughput, turnaround and resource utilization are produced and the simulation ends. (The simulation itself is a substantial but modular and thoroughly documented ALGOLW program.)

II. Input Parameters of the Simulation

The following parameters are required as input by the simulation:

- the network size (number of nodes)
- the total time to be simulated and the sampling interval (units are seconds)
- the memory size and limited acceptance threshold (if any) at each node (units are words of average 15 bits)
- the distributions and distribution parameters for processing times (input, execution and output) and memory needed for input, execution and output.
- the transmission speeds (capacities of communication lines) between the nodes (units are seconds per bit)
- the scheduling strategy or combination of strategies to be used
- a random number generator seed

A large number of experiments with the simulation are possible due to the quantity and variety of input parameters. Thus the choice of input parameters used in the simulation runs will be discussed.

It was decided to vary the network size from three to nine nodes in different experiments as a test of the consistency of the results. The total simulation time and sampling interval did not vary from their values of 10000 and 100 seconds respectively. The memory sizes at the nodes were set at 64K, to represent large minicomputers or medium-sized machines. The results obtained are therefore not dependent on the properties of individual nodes.

There are difficulties involved in selecting realistic distributions and parameters to characterize jobs in the

network. As the simulation is intended to model a research environment, a typical job was chosen to be a source deck of statements in a high-level language to be compiled and executed. A Poisson process was used to describe the arrival of jobs for processing, and an exponential distribution modeled the input, execution and output times [H1 pp.96-98]. Mean values for the latter three characteristics were taken from Brinch Hansen [B1 p.6]. The hyperexponential distribution was found to give a better fit to experimental data of arrival and processing times [B1 pp.195-198], but the load measure as defined necessitated the use of Poisson and exponential distributions. Values of the parameter λ_1 of the arrival process were found by experimentation (as explained in section 4.1 of Chapter IV).

The amount of memory requested by an average job is apportioned into three amounts. A certain amount is needed for input (to contain the card images). Execution requires at least this amount and usually considerably more (to be used for the object code generated). The memory requirement for output (the line images for the printer) is at most the execution requirement, so that the memory capacity of a node cannot be exceeded.

Transmission rates in the network have been restricted to two values - 750 bps (0.02 seconds per 15-bit word) and 1500 bps (0.01 seconds per 15-bit word). These values represent voice bandwidth facilities [A1 p.144]. The random

number generator seed is kept constant, ensuring identical job streams for all experiments with the same load and network size. A description of the random number generator used is given in [R2]. The input parameters of a sample simulation run (from the second set of the second series of experiments) are shown in Table III of the following section.

III. Output Statistics of the Simulation.

The output of the simulation consists of four pages of performance measurements. The output statistics produced by a sample run of the simulation are shown in Table III.

The input parameters are reproduced on the first page of the output. The second page contains the total number of jobs that completed processing and the total number of jobs that arrived at nodes in the network during the simulation. This is followed by a count of the longest intervals between events of the jobs which completed processing.

A breakdown by devices used of the jobs that finished and those being processed when the simulation ended, the average lengths of queues at each device, and the average turnaround times of jobs at each node are given on the third page. The fourth page contains statistics of resource utilization. The times during which devices were idle for lack of jobs to process and those in which they were idle for lack of memory needed to process jobs, are given as

percentages of the total simulation time. The sum of these times subtracted from 100 percent yields the percentage utilization. The percent memory usage and average percent of the acceptance threshold used (if the limited acceptance strategy is specfied) are given for each device.

INPUT PARAMETERS OF SIMULATION

NUMBER OF NODES: 6

TOTAL SIMULATION TIME: 10000.00 SAMPLING INTERVAL: 100.00

MEMORY SIZE AND LIMITED ACCEPTANCE THRESHOLD OF EACH NODE

NODE	MEMORY SIZE	L.A. THRESHOLD
1	64000	8960 (14.0%)
2	64000	8960 (14.0%)
3	64000	8960 (14.0%)
4	64000	8960 (14.0%)
5	64000	8960 (14.0%)
6	64000	8960 (14.0%)

POISSON ARRIVAL RATE OF JOBS AT EACH NODE

1	0.0167
2	0.0167
3	0.0167
4	0.0167
5	0.0167
6	0.0167

DISTRIBUTIONS AND PARAMETERS FOR PROCESSING TIMES AND MEMORY

INPUT TIME:	EXPONENTIAL	0.0500
EXECUTION TIME:	EXPONENTIAL	0.0167
OUTPUT TIME:	EXPONENTIAL	0.0330
INPUT MEMORY REQUEST:	UNIFORM	(1000.00, 4000.00)
EXECUTE MEMORY REQUEST:	UNIFORM	IMR + (2000.00, 6000.00)
OUTPUT MEMORY REQUEST:	UNIFORM	EMR - (0.00, 3000.00)

TRANSMISSION SPEEDS BETWEEN NODES (SEC/UNIT MEMORY)

	1	2	3	4	5	6
1	0.0000	0.0200	0.0100	0.0200	0.0100	0.0200
2	0.0200	0.0000	0.0200	0.0100	0.0100	0.0200
3	0.0100	0.0200	0.0000	0.0100	0.0200	0.0100
4	0.0200	0.0100	0.0100	0.0000	0.0200	0.0100
5	0.0100	0.0100	0.0200	0.0200	0.0000	0.0200
6	0.0200	0.0200	0.0100	0.0100	0.0200	0.0000

SCHEDULING STRATEGIES IN EFFECT

PRIOLOCAL
LIMACCEPT
OLDEST

TABLE III
Output of a Sample Simulation Experiment

NETWORK PERFORMANCE MEASUREMENTS

I. NETWORK THROUGHPUT

TOTAL NUMBER OF JOBS PROCESSED: 686
(SEQUENCE NUMBER OF LAST JOB GENERATED: 1005)

LONGEST TIME INTERVALS - NUMBER OF JOBS

JOBS EXECUTING AT INPUT NODE

INPUT START-GENERATION TIME	7
INPUT END-INPUT START	12
EXECUTION START-INPUT END	501
EXECUTION END-EXEC. START	76
OUTPUT START-EXECUTION END	28
OUTPUT END-OUTPUT START	15

JOBS EXECUTING AT REMOTE NODES

INPUT START-GENERATION TIME	0
INPUT END-INPUT START	0
TRANS. FOR EXEC. START-INPUT END	32
TRANS. FOR EXEC. END-TFE START	1
EXECUTION END-EXEC. START	6
TRANS. FOR OUTPUT START-EXEC. END	0
TRANS. FOR OUTPUT END-TFO START	6
OUTPUT START-TFO END	2
OUTPUT END-OUTPUT START	0

TABLE III (cont'd)
Output of a Sample Simulation Experiment

STATISTICS AT EACH NODE

JOBS PROCESSED BY EACH DEVICE

NODE	INPUT	LOCAL EXEC	REMOTE EXEC	OUTPUT
1	154	114	4	130
2	164	150	9	152
3	87	42	8	59
4	157	139	12	140
5	163	141	14	144
6	84	58	4	61

INPUT NODES OF REMOTELY EXECUTED JOBS

	1	2	3	4	5	6
1	0	3	4	3	3	3
2	0	0	0	2	2	0
3	4	4	0	5	5	1
4	0	1	0	0	3	0
5	0	0	2	1	0	0
6	0	1	2	1	1	0

AVERAGE QUEUE LENGTHS AT EACH NODE
(MEASURED AT END OF TIME INTERVALS)

NODE	INPUT QUEUE	EXECUTE QUEUE	OUTPUT WAIT QUEUE	OUTPUT QUEUE
1	4.34	12.08	0.03	0.65
2	0.45	3.33	0.64	0.30
3	26.14	20.50	0.00	0.06
4	0.41	3.91	0.98	0.51
5	0.44	6.02	0.11	0.30
6	24.84	15.54	0.00	0.16

II. AVERAGE TURNAROUND TIME OF JOBS ORIGINATING AT EACH NODE

1	625.71
2	323.48
3	852.87
4	391.73
5	466.15
6	594.39

TABLE III (cont'd)
Output of a Sample Simulation Experiment

III. DEVICE UTILIZATION

% IDLE TIME (NO JOBS TO PROCESS)

NODE	INPUT DEV	CPU DEV	OUTPUT DEV
1	39.56	0.92	59.00
2	67.82	8.45	56.38
3	15.72	2.30	27.37
4	72.38	5.05	57.37
5	68.96	0.23	53.66
6	26.97	2.07	34.75

% IDLE TIME DUE TO LACK OF AVAILABLE MEMORY

NODE	INPUT DEV	CPU DEV	OUTPUT DEV
1	26.17	24.70	0.00
2	1.13	0.43	0.00
3	67.58	63.56	54.53
4	0.00	0.00	0.00
5	3.23	1.62	0.00
6	58.52	57.83	47.95

% UTILIZATION (100% - IDLE TIMES(%))

NODE	INPUT DEV	CPU DEV	OUTPUT DEV
1	34.26	74.36	40.99
2	31.03	91.10	43.61
3	16.69	34.13	18.09
4	27.61	94.94	42.62
5	27.79	98.14	46.33
6	14.49	40.09	17.28

MEMORY USED AT EACH NODE (AVERAGED OVER INTERVALS)

NODE	% MEMORY USED	% L.A. THRESHOLD USED
1	68.33	6.85
2	40.92	62.97
3	84.76	3.11
4	42.28	46.02
5	45.60	18.49
6	79.84	4.82

SUMMARY OF PERFORMANCE MEASUREMENTS

THROUGHPUT (%):	68.25
REMOTE JOBS/TOTAL JOBS PROCESSED (%):	6.85
NETWORK AVERAGE TURNAROUND TIME:	542.39
NETWORK AVERAGE CPU UTILIZATION (%):	72.13

TABLE III (cont'd)
Output of a Sample Simulation Experiment